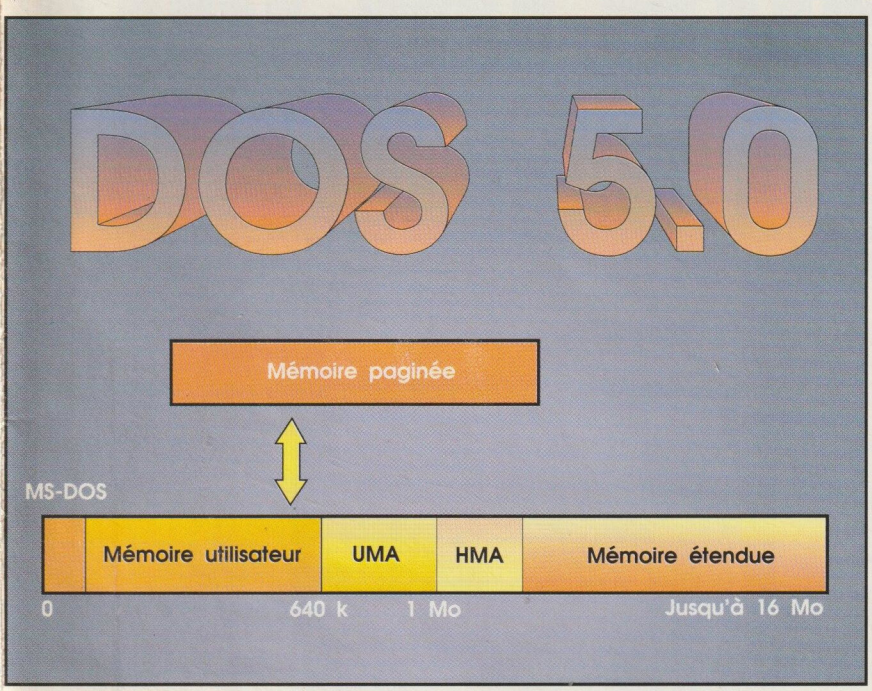
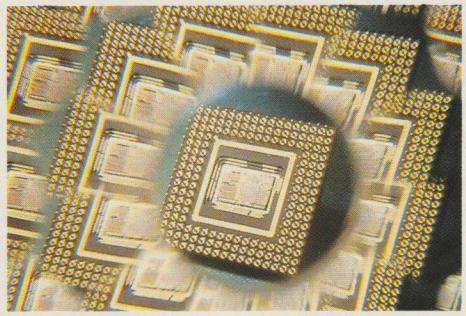
INFOPC
langages & systèmes

OCTOBRE 1991 - Nº 4
SUPPLÉMENT
DÉVELOPPEURS
DU Nº 75 D'INFOPC



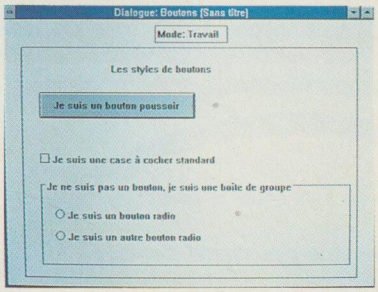
MS-DOS 5.0

De nouvelles fonctions facilitent le chargement des drivers et TSR en mémoire haute



INITIATION WINDOWS 3

Seconde partie : Fenêtres de contrôle et support MDI



LES COPROCESSEURS DU 386DX ET i486

Voyage au cœur des unités de traitement mathématique

ET AUSSI...

- **LA GÉOMÉTRIE FRACTALE**
- **PC-LINT**
- **INSTALL**

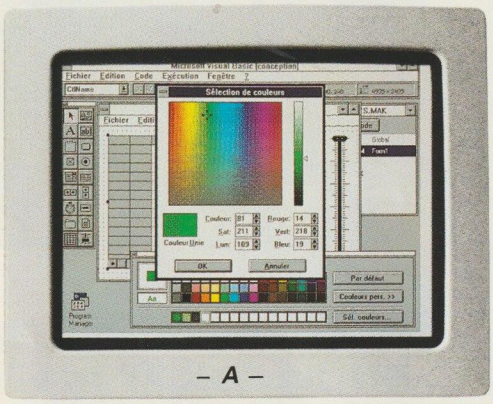
LA DÉTECTION DES CONTOURS D'IMAGES

Présentation des approches mathématiques Gradient et Laplacien

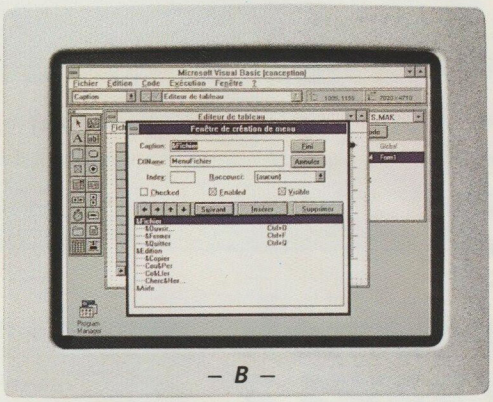
NE PEUT ÊTRE VENDU SÉPARÉMENT

Microsoft Visu Windows : il suffit pour faire de votr en programmatio

Avec Visual Basic 1.0, le monde de la programmation sous Windows ne sera plus jamais comme avant. Ce qui nécessitait parfois des mois de travail va maintenant se compter en journées. Que vous soyez développeur ou que vous souhaitiez vous lancer dans la programmation sous Windows pour vos besoins spécifiques, Microsoft Visual Basic pour Windows est, de loin, l'outil le plus facile et le plus rapide d'emploi. Conséquence, c'est le plus productif.



– A –



– B –

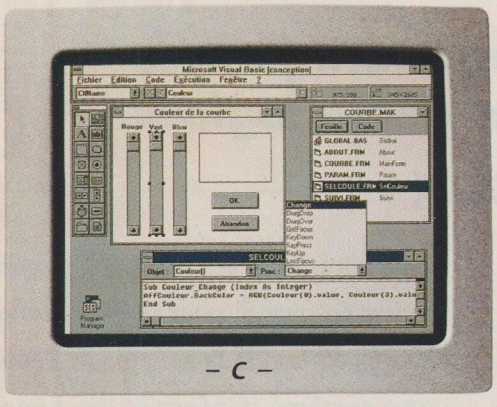
A. **Dessinez votre interface avec la souris.** Avec Visual Basic, vous créez l'interface utilisateur de votre application graphiquement sans écrire une seule ligne de code. Vous dessinez vos fenêtres en sélectionnant l'un des 15 types de contrôles disponibles dans la boîte à outils de Visual Basic. Pour les rendre plus attrayantes, vous pouvez y incorporer facilement des images que vous choisissez dans une bibliothèque de plus de 400 icônes fournie avec Visual Basic. Un éditeur d'icônes vous permet même de les modifier ou d'en créer de nouvelles.

B. **Créez vos menus visuellement.** Vous pouvez créer vos menus très facilement et associer chaque fonction à un accélérateur clavier grâce à la fenêtre de création de menus.

C. **Programmez à l'aide du langage le plus accessible de la micro-informatique.** La programmation est ensuite réservée aux traitements internes de vos applications. Elle s'effectue dans un

ıal Basic pour
de quelques clics
'e micro un crack
n sous Windows.

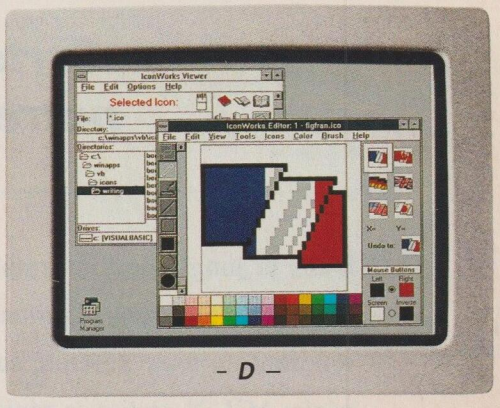
langage Basic puissant et structuré, issu de QuickBasic et doté d'extensions orientées objets pour accéder facilement aux fonctionnalités de Windows. Vous saisissez votre code directement sous Windows en le rattachant très simplement aux éléments de votre interface graphique.



– C –

***D.* Exécutez et mettez au point votre application instantanément sous Windows.** La mise au point de vos programmes s'effectue directement sous Windows avec la possibilité d'exécuter le code pas à pas et de visualiser des variables.

Vous pouvez même modifier votre code pendant son exécution et relancer l'exécution au point d'arrêt ! Une fois la mise au point terminée, vous créez de véritables applications exécutables Windows qui sont diffusables sans run-time.



– D –

Accédez facilement à toute la puissance de Windows. Visual Basic est la plate-forme d'intégration idéale : la gestion du presse-papier, de l'impression, du lancement d'autres applications et l'échange dynamique de données (DDE) se programment très simplement. Vous pouvez également compléter votre application avec des bibliothèques dynamiques (DLLs) fournies par des éditeurs ou développées à l'aide de langages comme Microsoft C PDS 6.0, Microsoft FORTRAN PDS 5.1, Microsoft COBOL PDS 4.5 ou Microsoft Macro-Assembler PDS 6.0. Dans toutes les situations, vous pourrez construire votre solution. Microsoft Visual Basic est disponible en version française au prix de 1990 F HT.*

Pour toutes informations complémentaires, appelez le Service Clients au (1) 69 86 46 46 ou tapez 3616 code Microsoft.

Microsoft, 91957 Les Ulis cedex et à Infomart La Défense - Stand N° 276 niveau 2.

Microsoft®

LES LOGICIELS QUI DONNENT DES AILES.

*soit 3 711,35 F TTC prix public conseillé au 1/09/91.

Service lecteurs n° 1 (dernière page)

ORENOK PHOTO IMAGE BANK

Certains n'adopteront jamais notre atelier de génie logiciel.

AGL-X

Dans la jungle de l'informatique de gestion, il y a ceux qui font la grimace et ceux qui se reposent en toute sérenité sur un assistant de génie: AGL-X, atelier de génie logiciel.

Dès son arrivée dans votre entreprise AGL-X s'intègre dans votre existant: vous capitalisez vos investissements passés. AGL-X couvre toutes les phases de vos futurs projets: un seul outil pour tous vos besoins.

Toutes les fonctions d'AGL-X (prototypage, L4G, générateur GAP ou COBOL, générateur de documentation, sortie graphique...) vous aident à atteindre vos objectifs: produire vite un résultat de qualité pour un coût minimum. Cet environnement est la garantie de l'évolutivité de votre système d'information: vous réduisez enfin vos coûts de maintenance. Conçu pour l'IBM AS 400, AGL-X exploite au mieux les qualités de votre système informatique et respecte les recommandations AD/CYCLE.

Alors, si vous préférez la rentabilité, la convivialité et la sécurité à la jungle informatique, prenons le temps d'en parler ensemble...

Contactez M. Dacalor: (1) 34 65 93 70.

PRISME
CONCEPTEUR - DÉVELOPPEUR
2-4, rue Jean-Baptiste Huet - 78353 Jouy en Josas Cedex

Service lecteurs n° 2 (dernière page)

Sommaire

- 6 -

Re-sources

Une sélection des nouveaux outils de développement

- 10 -

Apprendre Windows

Deuxième partie de cette initiation à la programmation sous l'environnement graphique Microsoft Windows 3 : fenêtres de contrôle et support MDI

- 38 -

MS-DOS 5.0

Description de fonctions jusqu'alors non documentées, et mise en oeuvre de celles donnant accès aux UMB

- 54 -

Les coprocesseurs arithmétiques du 386DX et i486

Présentation des avantages et inconvénients de ces auxiliaires de l'unité centrale

- 62 -

La détection des contours d'images

Les mathématiques au service de la reconnaissance des formes, première étape sur le chemin de la vision artificielle.

- 72 -

La géométrie fractale

La théorie des fractales permet d'atteindre des taux de compression élevés pour le stockage des images.

- 78 -

PC-Lint

Une solution logicielle pour déceler les "bugs" éventuels dans le source d'un programme en C.

- 80 -

Install

Un outil pour réussir le programme d'installation d'une application.

La disquette de la rédaction d'Info PC du mois de novembre contient les listings de Langages & Systèmes

Editorial

Dix ans se sont écoulés depuis la commercialisation du PC. Ce produit a connu maints perfectionnements dans tous les domaines (stockage, affichage, puissance de calcul, ...), tout en voyant son prix baisser graduellement. Les langages ont connu parallèlement une profonde mutation. En effet, il y a dix ans, les développeurs ne disposaient, outre l'assembleur, que de quatre compilateurs : Basic, Fortran, Pascal et Cobol (le C ne s'imposera qu'en 1986). Ils étaient relativement rudimentaires (pas d'éditeur pleine page, jeu d'instructions restreint, compilation en plusieurs passes, etc.), et leurs prix élevés les réservaient à une petite élite.

Aujourd'hui, ces langages se sont démocratisés et ont gagné en puissance et en ergonomie. Mais cet essor, somme toute prévisible, a été accompagné d'une révolution avec l'apparition de la programmation orientée objet. On manipule désormais des objets qui peuvent être aussi bien du code exécutable, des images, des sons, etc. Cette nouvelle approche de la programmation va accélérer la migration vers le tout graphique, notamment sous Windows, car le développeur n'aura plus à se soucier de tous les aspects de l'interface graphique (ascenseurs, boutons, etc.), ceux-ci étant disponibles sous forme d'objets.

Et dans dix ans ? Gageons que s'il ne reste qu'un seul langage, ce sera le Basic ! Le Beginner's All purpose Symbolic Instruction Code a connu de nombreuses métamorphoses, dont la dernière en date est le Visual Basic (une version SQL est d'ailleurs annoncée), et demeure le moins contraignant des langages. Il permet en effet des digressions (les fameuses variables qui sont inventées au fur et à mesure des besoins), tout en offrant les garanties nécessaires pour l'écriture des programmes les plus structurés. Tout le désigne en fait pour devenir le langage universel par lequel tout développeur devra passer.

Yves Guittard

Re-sources

Un complément P.O.O. sur Mac

ACI commercialise depuis peu un nouvel éditeur orienté objet. S'adressant aux professionnels de la P.O.O. sur Macintosh, Object Master, par de nombreux outils, améliore les performances et la convivialité de leur plate-forme de développement.

OBJECT MASTER

• Editeur orienté objet • Gestion projet polyvalente et compatible • Analyseur structurel et syntaxique • Attributs couleur, style, protection • Edition multiple • Arborescence objet et gestion mémoire • Génère code assembleur•

Object Master peut traiter, en utilisant les "Apple events" du Système 7, les fichiers sources créés en Macintosh Programmer's Workshop, MPW, C/C++, en THINK Pascal et C ainsi qu'en Modula2, exécuter ResEdit et ISDN. D'abord, il effectue une analyse structurelle et syntaxique des programmes concernés par un projet et crée un fichier exécutable s'il n'existe pas. Ensuite, les principales données comme la segmentation, les ressources, les mots clés, les classes ou les méthodes, sont traduites en mode graphique. Object Master gère durant l'édition les attributs de couleur ou de style spécifiques aux fonctions, variables, ce qui facilite la relecture du texte et chaque élément structurel est accessible séparément. Cet outil crée aussi une arborescence objet dynamique mettant en évidence l'héritage et le contenu des classes. Enfin, pour faciliter le travail d'équipe sur un même projet, Object Master gère la protection en écriture des modules en cours de mise au point par une personne. Une option génère même, ligne par ligne, l'équivalent du programme en assembleur. La gestion des segments de code est simplifiée par une planification de la mémoire à répartir. Un menu dédié aux macros permet d'insérer dans le programme la base des expressions conditionnelles pré-définies comme un "tant que".

Développement objet en kit sous Windows

Level5 Object est un outil de programmation destiné aux professionnels du développement orienté objet sous Windows. L'ensemble se compose entre autres d'un générateur de système expert avec module d'aide à la décision, d'un éditeur de fichiers sources orienté objet, d'un debugger et d'une base de données relationnelle. Dans un environnement souple, convivial et multifenêtré, l'utilisateur peut construire l'analyse de la hiérarchie correspondant aux diverses étapes de son application, en indiquant les buts des procédures mises en place, par exemple, acheter ou vendre, avec à chaque fois la classe et le domaine. Une arborescence des principaux événements apparaît et chaque case peut être éditée pour y rentrer les expressions et paramètres. L' utilisation du langage, déjà très "L4G", est simplifiée par l'existence de macros. Un éditeur de fenêtres graphiques permet alors de personnaliser l'aspect du traitement et le formatage des résultats (zones de saisie, boutons, menu...).

LEVEL5 OBJECT

•Outil programmation Windows • Editeur orienté objet avec debugger • Générateur système expert • Base de données relationnelle • Éditeur de fenêtres et historique des commandes •

Compression et décompression

PKWare met à disposition des programmeurs deux puissants algorithmes, Implode et Explode, écrits en assembleur optimisé. Pour chaque donnée, le programme de compression effectue une analyse dans une fenêtre mobile sur une partie prédéterminée du fichier et répertorie les éventuelles redondances, pour ensuite les codifier dans un autre fichier.

PKWARE

•Compression Library • Deux algorithmes en Assembleur optimisé • Gestion des I/O, erreurs et des arguments • Fenêtre de recherche mobile paramétrable • Compatibles principaux langages, sous DOS, Windows, OS/2 • Pour tout PC de la famille des 80 X 88 • Mode réel ou protégé •

Les diverses fonctions assurent un bon contrôle des entrées-sorties, des erreurs et des arguments. Les données compressées, en mode ASCII ou binaire, peuvent être traitées sur disque, en mémoire ou redirigées vers un port série. La taille du dictionnaire, fenêtre mobile de recherche des occurrences redondantes, est paramètrable de 1024 à 4096 octets et détermine la rapidité du traitement. Compatible avec tous les CPU 80 X 86 en mode réel ou protégé, la bibliothèque de compression PKWare est utilisable sous DOS, Windows, OS/2, pour des applications en Assembler, C, Pascal, Clipper, Fortran, Basic.

Matthieu Thfoin

Software Development

Le salon Software Development organisé par Capric se déroulera du 15 au 17 octobre au CNIT Paris La Défense. Il réunira les principaux acteurs de l'industrie logicielle, aussi bien dans le domaine des progiciels que dans celui des outils de développement. Les visiteurs y rencontreront notamment Microsoft, Ingres, Oracle, Memsoft, etc. Des conférences élaborées et animées par le CXP auront également lieu et feront appel à des témoignages d'utilisateurs de grandes sociétés.

Mardi 15 octobre

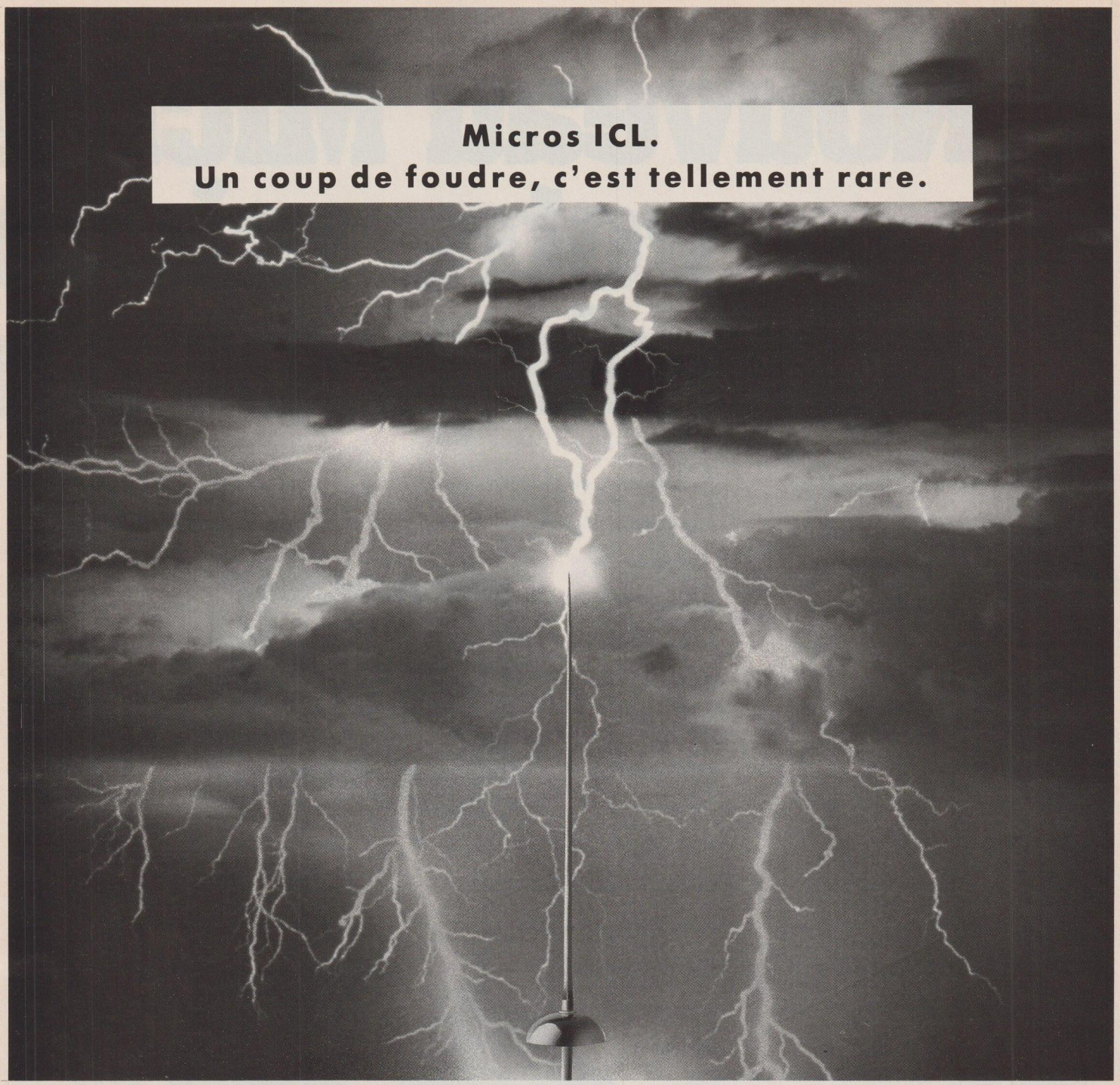
9h30	Conférence inaugurale
14h30	Conduite des projets informatiques
14h30	Techniques de restructuration d'applications et de Reverse Engineering

Mercredi 16 octobre

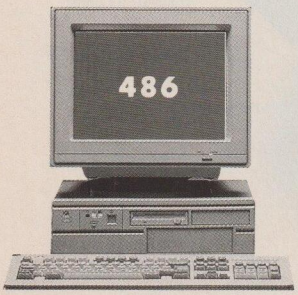
9h30	Méthodes et outils d'élaboration du schéma directeur
9h30	Maîtrise de la qualité en informatique de gestion
14h30	Méthodes de conception et de développement du système d'information
14h30	Techniques de test, de recettes et de mise en exploitation

Jeudi 17 octobre

9h30	Construction d'interfaces applicatives graphiques et techniques de remodelage ergonomiques (re-vamping)
9h30	Développement d'applications et SGBD (généralisation du relationnel)
14h30	Conception de développement d'applications réparties (traitement coopératif et architecture client / serveur)
14h30	Mise en œuvre d'un atelier de génie logiciel (développement sur grand système ou sur architecture de réseau local)



Service lecteurs n° 3 (dernière page)



Avec ICL, certaines idées reçues en matière de micro informatique risquent d'en prendre un sacré coup. Alliance de la puissance et de la beauté, les micros ICL ont tout pour vous séduire : gamme complète, design extra-compact, écran VGA haute définition anti-reflet, clavier ergonomique et agréable.

Et sur le plan technique, c'est le grand frisson garanti. Totalement compatible MS-DOS, la gamme ICL (du 286 au 486 sous UNIX) vous ouvre l'accès aux bases de données de votre entreprise et du monde extérieur. Vitesse d'horloge ultrarapide, disques durs de grande capacité, mémoires extensibles, toutes ces performances pour un rapport qualité/prix exceptionnel.

Après de tels arguments, comment ne pas avoir le coup de foudre pour les micros ICL !

ICL

24, avenue de l'Europe, 78140 Vélizy.
Téléphone : 34.65.80.70.

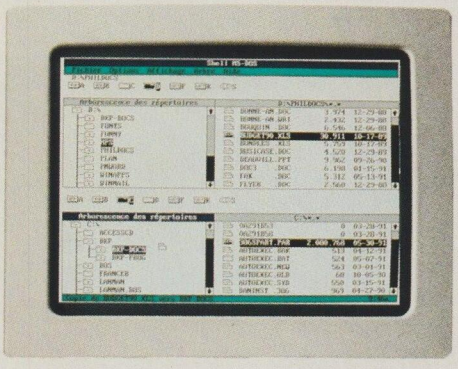
A LA POINTE DE L'INFORMATIQUE

Nouveau Micros votre micro rajeunir pour mo

Rajeunir ! Retrouver l'énergie et la souplesse de la jeunesse, c'est ce que va pouvoir réaliser votre plus fidèle allié : votre micro. C'est pour lui que Microsoft lance aujourd'hui la Mise à Jour MS-DOS 5.0 conçue pour les systèmes d'exploitation des micros IBM PC, PS/2 et compatibles.

Bien que la mémoire conventionnelle de votre PC soit loin d'être défaillante, lui en donner plus lui permettra d'accueillir des applications à court de mémoire ou de repousser les limites de taille de fichiers. Sur un PC 80286 ou i386™, MS-DOS 5.0 augmente ainsi la mémoire disponible d'au moins 45 Ko dans la plupart des configurations.

MS-DOS 5.0 dispose d'un centre de contrôle totalement renouvelé à partir duquel peuvent être exécutées toutes les commandes DOS. Ce "Shell" MS-DOS, avec son interface à menus déroulants et son support complet de la souris, constitue un environnement beaucoup plus convivial que la célèbre ligne de commande DOS.

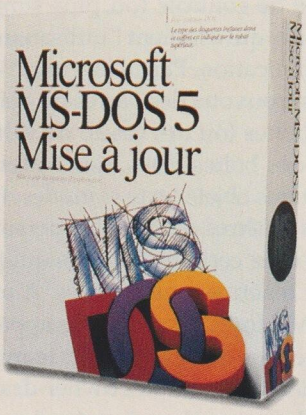


Avec MS-DOS 5.0, Microsoft s'est attaché à rendre les manipulations les plus courantes à la fois simples et performantes. Ainsi, un système complet d'aide est accessible à tout moment, soit par le menu dans le "Shell," soit en une seule instruction à la ligne de commande. Dans le Gestionnaire de Fichiers, répertoires et fichiers se présentent sous forme d'une arborescence d'icônes. Ainsi, pour déplacer un fichier, il suffit de déplacer son icône avec la souris ou d'ouvrir une simple boîte de dialogue. Le Gestionnaire de Programmes permet de regrouper les applications en fonction de votre propre organisation.

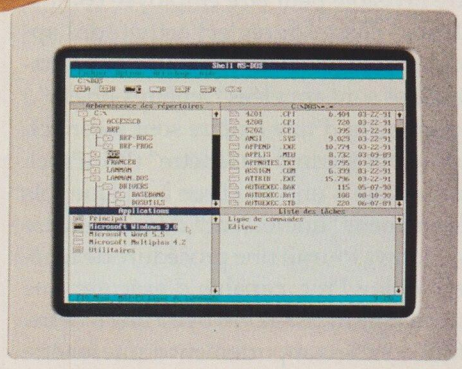
oft MS-DOS 5.0 : va se sentir ins de 790 F. ttc.*

Et, avec la nouvelle fonction de Commutation de Tâches, vous pouvez charger puis mettre en attente plusieurs applications sans qu'il soit nécessaire de les fermer pour passer de l'une à l'autre.

L'ancien éditeur de ligne "Edlin" est remplacé par un nouvel éditeur pleine page à menus déroulants "Edit." Succédant à l'interpréteur GW-Basic, le MS-DOS QBasic est un langage de programmation Basic moderne qui offre rapidité et simplicité aux utilisateurs intéressés par la programmation structurée.



Pour vous garantir contre toutes les erreurs de manipulation, MS-DOS 5.0 dispose également d'utilitaires sophistiqués tels que les commandes "Undelete" (récupération de fichiers effacés), "Unformat" (annulation de formatage de disque), ou encore la fonction de reformatage rapide d'une disquette.



Enfin, la Mise à Jour MS-DOS 5.0 est si facile à installer (quelle que soit votre version antérieure de MS-DOS à partir de la 2.10) que vous pourrez remettre à niveau chaque PC de votre parc en un clin d'œil. L'installation peut même se faire à travers le réseau lors d'une session de connexion. De plus, le package de Mise à Jour dispose d'une procédure de désinstallation vous permettant de retrouver, si besoin était, votre configuration initiale équipée de votre ancienne version de DOS. Grâce à un test intensif de plusieurs mois sur plus de 7 000 sites à travers le monde, la Mise à Jour MS-DOS 5.0 est le logiciel le plus testé de l'histoire de l'informatique avant sa mise sur le marché. Toutes les configurations matérielles et logicielles possibles ont été envisagées pour offrir une fiabilité maximale à votre configuration.

La Mise à Jour Microsoft MS-DOS 5.0 est maintenant disponible chez votre revendeur habituel.

Pour toutes informations complémentaires, appelez le Service Clients au (1) 69 86 46 46 ou tapez 3616 code Microsoft.

Microsoft, 91957 Les Ulis cedex et à Infomart La Défense - Stand N° 276 niveau 2.

Microsoft®

LES LOGICIELS QUI DONNENT DES AILES.

* Prix public conseillé de la mise à jour au 1/09/91 - i386 est une marque déposée d'Intel Corporation. IBM est une marque déposée d'International Business Machines.

Service lecteurs n° 4 (dernière page)

APPRENDRE WINDOWS

2° partie : Fenêtres de contrôle et support MDI

MARC ADLER

Commençons par réviser le sens du mot "fenêtre". Une fenêtre est un objet (possédant des données) auquel sont associées certaines propriétés. Elle peut occuper une zone de l'écran, bien que cela ne soit pas obligatoire. Elle peut demeurer invisible durant toute la durée d'exécution de l'application. (Une fenêtre invisible peut être employée comme un "agent de la circulation" chargé de régler l'activité de l'application et de traiter les messages. Dans Presentation Manager, sous OS/2, ces fenêtres sont appelées "fenêtres-objets".)

L'aspect et le comportement d'une fenêtre sont essentiellement déterminés par sa "procédure de fenêtre" (WinProc). Windows envoie un message à une WinProc à chaque fois qu'il se produit un événement "intéressant" dans la fenêtre qui lui est associée. Windows fournit une procédure de fenêtre "par défaut", DefWindowProc, capable d'exécuter une action par défaut pour chaque message envoyé à une fenêtre. DefWindowProc définit donc le comportement d'une fenêtre générique. Pour modifier le comportement ou l'aspect d'une fenêtre - ou d'un objet contenu dans cette fenêtre - certains de ces messages doivent être gérés par le code de la WinProc plutôt que par la DefWindowProc.

FIGURE 1
Styles des contrôles de Windows

Scrollbar	Barre de défilement horizontale ou verticale
Edit	Champ d'édition monoligne ou multiligne
Listbox	Boîte liste monosélection ou multisélection
Combobox	Boîte combo simple, déroulante, ou à liste déroulante
Static	Cadre; chaîne justifiée à gauche, centrée ou justifiée à droite
Button	Bouton poussoir, bouton radio, case à cocher, boîte de groupe
MDIClient	Fenêtre cliente MDI ("Multiple Document Interface"); Remarque: il ne s'agit pas d'une fenêtre de contrôle

Par exemple, vous pouvez traiter des messages souris pour indiquer à un contrôle affiché comment il doit réagir lorsque vous cliquez sur lui. Imaginons l'implémentation d'un contrôle représenté par un interrupteur personnalisé, qui change de position à chaque fois que vous cliquez dessus. Lorsque l'on presse le bouton de la souris, le programme doit déterminer la position de l'interrupteur (haut ou bas) puis le redessiner dans la position inverse. Pour cela, il faut intercepter le message WM_LBUTTONDOWN, inverser la valeur de la variable d'état interne à ce contrôle, puis redessiner le contrôle, en utilisant probablement une image bitmap différente pour afficher la nouvelle position. Vous pouvez donc réécrire le code de traitement de certains messages pour amener une fenêtre à se comporter selon vos souhaits.

Les contrôles

Les fenêtres de contrôle (ou plus simplement "contrôles") sont des objets qui assistent l'utilisateur dans son interaction avec une application. Parmi les contrôles prédéfinis de Windows, nous trouvons des barres de défilement, des boutons radio, des boîtes (ou des cases) de validation, des boutons-poussoirs, des boîtes listes, des boîtes combo, des champs d'édition et des objets texte statiques. Il est possible de créer ses propres fenêtres de contrôle personnalisées, ou bien de les acheter chez certains éditeurs qui proposent des bibliothèques de fenêtres toutes prêtes. A mesure que grandit le succès de Windows, nous voyons apparaître de plus en plus de bibliothèques de ce genre sur le marché, surtout grâce à l'introduction des contrôles owner-draw dans Windows 3.0.

Il existe deux manières de créer des contrôles. Il est possible d'utiliser la fonction CreateWindow (ainsi que la fonction CreateWindowEx de Windows 3.0) ou encore d'employer l'éditeur de dialogue et définir des contrôles comme étant des composants d'une boîte de dialogue. Bien que cette seconde méthode soit la plus courante, nous commencerons par examiner CreateWindow.

Le premier article de cette série (dans notre précédent numéro) a montré comment utiliser CreateWindow pour

Notre série d'articles "Apprendre Windows" se propose d'explorer la programmation de l'environnement graphique Microsoft Windows version 3.0. Dans le numéro précédent, nous avons présenté la gestion des événements, l'envoi des messages, l'initialisation d'une application, les fichiers ressource, les systèmes de menu et les raccourcis. De plus, nous avons proposé le schéma de notre application d'exemple : un portefeuille de titres. Le présent article traite, quant à lui, des types de fenêtres de contrôle supportés par Windows et poursuit le développement de l'application d'exemple en y ajoutant un support pour l'architecture MDI (Multiple Document Interface).

construire la fenêtre principale de notre application d'exemple. Deux des paramètres de la fonction CreateWindow présentent un intérêt particulier : le nom de la classe de fenêtre et le handle de la fenêtre mère.

Dans l'environnement Windows, les fenêtres peuvent avoir une relation mère-fille. Les fenêtres filles apparaissent à l'intérieur de la fenêtre propriétaire, appelée fenêtre mère. L'affichage des fenêtres filles est limité à la zone cliente de la fenêtre mère : c'est-à-dire la zone principale, initialement vide, qui ne contient pas la barre de légende, ni la bordure de fenêtre, ni la barre de menu ou les barres de défilement. Les parties des fenêtres filles qui dépassent la bordure de la zone cliente ne sont pas affichées sur l'écran.

La fille hérite de certaines caractéristiques du comportement de sa mère. Si une fenêtre mère est cachée ou détruite, ses filles sont également cachées ou détruites. Il en va de même pour les filles des fenêtres filles, et ainsi de suite dans la hiérarchie des fenêtres. Lorsqu'une fenêtre est déplacée, toutes ses descendantes sont déplacées avec elle. Dans le cas des fenêtres de contrôle, l'aspect le plus important de la relation mère-fille est le suivant : lorsque certains événements concernent une fenêtre de contrôle, cette dernière informe sa fenêtre mère en lui envoyant un message WM_COMMAND. (Une barre de défilement envoie un message WM_VSCROLL ou WM_HSCROLL.)

Dans notre application boursière, la fenêtre principale n'avait pas de parent. On peut le constater en examinant le paramètre hParent de la fonction CreateWindow employée pour créer la fenêtre. Si le paramètre hParent a pour valeur NULL, la fenêtre n'a pas d'ancêtre - on parle alors de fenêtre de niveau supérieur. (En réalité, la fenêtre de niveau supérieur est une fille de la fenêtre du bureau, mais cela n'est jamais une préoccupation pour le programmeur Windows.)

Les fenêtres de contrôle sont presque toujours des fenêtres filles. Par conséquent, lorsque vous employez CreateWindow pour créer un contrôle, vous devez spécifier le handle d'une fenêtre mère.

Windows prédéfinit un certain nombre de classes de fenêtres de contrôle. Une liste des noms de classes de contrôle et des types de contrôles disponibles est donnée dans la Figure 1.

Utilisation de CreateWindow

Lorsque vous employez CreateWindow pour créer un contrôle d'une classe prédéfinie, vous devez fournir comme nom de classe une des chaînes de la Figure 1. Il est souvent nécessaire de spécifier un ou plusieurs drapeaux pour le troisième argument de l'appel à CreateWindow. L'appel suivant crée un bouton-poussoir :

```
hButton = CreateWindow("Button",
                "&OK",
                BS_PUSHBUTTON | WS_CHILD |
WS_TABSTOP,
                100, 75, 32, 16,
                hWndMain,
                IDOK,
                hInstance,
                (LPSTR) NULL);
```

Cet appel présente plusieurs caractéristiques intéressantes. On remarquera d'abord l'esperluète (caractère &) au début de la chaîne du bouton : elle spécifie un raccourci (ou accélérateur). Ensuite, le drapeau BS_PUSHBUTTON est un des bits de style, qui indique à Windows que l'on souhaite créer un bouton-poussoir. La Figure 2 contient une liste complète des drapeaux utilisables pour chaque classe de fenêtres. Le drapeau WS_CHILD précise à Windows qu'il s'agit d'une fenêtre fille et qu'il faut employer hWndMain comme parent.

FIGURE 2
Styles de Windows

STYLE	DESCRIPTION
DS_LOCALEDIT	Spécifie que le contrôle d'édition dans la boîte de dialogue utilisera de la mémoire dans le segment de données de l'application.

.../...

	Par défaut, tous les contrôles d'édition des boîtes de dialogue utilisent de la mémoire en dehors du segment de données de l'application. Cette caractéristique peut être supprimée en ajoutant le drapeau DS_LOCALEDIT à la commande STYLE de la boîte de dialogue. Si ce drapeau n'est pas employé, les messages EM_GETHANDLE et EM_SETHANDLE ne doivent pas non plus être utilisés puisque la zone de stockage mémoire du contrôle ne se trouve pas dans le segment de données de l'application. Cette caractéristique n'affecte pas les contrôles d'édition créés en dehors des boîtes de dialogue.
DS_MODALFRAME	Crée une boîte de dialogue avec un cadre modal qui peut être combiné à une barre de titre et un menu système en spécifiant les styles WS_CAPTION et WS_SYSMENU.
DS_NOIDLEMSG	Supprime les messages WM_ENTERIDLE que Windows enverrait normalement à la fenêtre propriétaire de la boîte de dialogue lorsque cette boîte est affichée.
DS_SYSMODAL	Crée une boîte de dialogue modale système.
WS_BORDER	Crée une fenêtre possédant une bordure.
WS_CAPTION	Crée une fenêtre possédant une barre de titre (sous-entend l'activation du style WS_BORDER). Ce style ne peut pas être employé conjointement avec le style WS_DLGFRAME.
WS_CHILD	Crée une fenêtre fille. Ne peut pas être employé conjointement avec le style WS_POPUP.
WS_CHILDWINDOW	Crée une fenêtre fille ayant le style WS_CHILD
WS_CLIPCHILDREN	Exclut la zone occupée par les fenêtres filles lors d'un affichage à l'intérieur de la fenêtre mère. Spécifié lors

.../...

	de la création de la fenêtre mère.
WS_CLIPSIBLINGS	Coupe (limite) les fenêtres filles relativement les unes aux autres; autrement dit, lorsqu'une fenêtre fille reçoit un message d'affichage, le style WS_CLIPSIBLINGS coupe toutes les autres fenêtres filles partiellement superposées, afin qu'elles ne débordent pas sur la fenêtre qui vient de recevoir le message. (Si WS_CLIPSIBLINGS n'est pas spécifié et que des fenêtres filles se recoupent, l'affichage dans la zone cliente d'une fenêtre fille peut éventuellement déborder dans la zone cliente d'une autre fenêtre fille.) Ce style doit être uniquement employé en conjonction avec le style WS_CHILD.
WS_DISABLED	Crée une fenêtre initialement inactive.
WS_DLGFRAME	Crée une fenêtre avec une double bordure mais sans titre.
WS_GROUP	Spécifie le premier contrôle d'un groupe dans lequel l'utilisateur peut se déplacer d'un contrôle au contrôle voisin à l'aide des touches fléchées. Tous les contrôles définis ensuite appartiennent au même groupe. Le prochain contrôle possédant le style WS_GROUP marque la fin du groupe et le début du groupe suivant (autrement dit, un groupe se termine là où commence le suivant). Seules les boîtes de dialogue peuvent employer ce style.
WS_HSCROLL	Crée une fenêtre possédant une barre de défilement horizontale.
WS_ICONIC	Crée une fenêtre initialement réduite en icône. A utiliser uniquement avec le style WS_OVERLAPPED.
WS_MAXIMIZE	Crée une fenêtre agrandie à sa taille maximale.
WS_MAXIMIZEBOX	Crée une fenêtre ayant un

.../...

```
                bouton "Agrandissement"
                (case de mise en taille
                maximale, en haut à droite).
WS_MINIMIZE     Crée une fenêtre réduite en
                icone.
WS_MINIMIZEBOX  Crée une fenêtre ayant un
                bouton "Réduction"
                (case de mise en icône, en
                haut à droite).
WS_OVERLAPPED   Crée une fenêtre superposable
                ("overlapped") possédant un
                titre et une bordure. Il
                s'agit généralement d'une
                fenêtre principale.
WS_OVERLAPPEDWINDOW     Crée une fenêtre
                "overlapped" ayant les styles
                WS_OVERLAPPED, WS_CAPTION,
                WS_SYSMENU, WS_THICKFRAME,
                WS_MINIMIZEBOX et
                WS_MAXIMIZEBOX. Il s'agit
                normalement d'une fenêtre
                principale.
WS_POPUP        Crée une fenêtre pop-up. Ne
                peut pas être employé
                conjointement avec le style
                WS_CHILD.
WS_POPUPWINDOW  Crée une fenêtre popup
                ayant les styles WS_BORDER,
                WS_POPUP et WS_SYSMENU. Le
                style WS_CAPTION doit être
                combiné avec le style
                WS_POPUPWINDOW pour que le
                menu système soit visible.
WS_SYSMENU      Crée une fenêtre possédant un
                bouton de menu système dans la
                barre de titre (case de menu
                système, en haut à gauche).
                Utilisé uniquement avec les
                fenêtre possédant une barre de
                titre.
WS_TABSTOP      Spécifie un des contrôles
                parmi lesquels l'utilisateur
                peut se déplacer à l'aide de
                la touche de tabulation.
                La touche de tabulation
                déplace l'utilisateur vers le
                prochain contrôle possédant le
                style WS_TABSTOP. Seules les
                boîtes de dialogue emploient
                ce style.
WS_THICKFRAME   Crée une fenêtre possédant
                une bordure épaisse utilisable
                pour redéfinir la taille de la
                                        .../...
```

```
                fenêtre.
WS_VISIBLE      Crée une fenêtre initialement
                visible. Ce style s'applique
                aux fenêtres overlapped et
                aux fenêtre pop-up. Pour les
                fenêtres overlapped, le
                paramètre Y est employé comme
                paramètre de la fonction
                ShowWindow.
WS_VSCROLL      Crée une fenêtre possédant une
                barre de défilement verticale.
```

L'on notera également que le paramètre hMenu a pour valeur IDOK. Cette valeur, définie dans WINDOWS.H, est employée généralement pour les boutons OK. Quand vous créez une fenêtre fille, la valeur du paramètre hMenu doit être l'identificateur du contrôle de la fenêtre, et pas un handle de menu. Windows surcharge les champs du handle de menu et de l'identificateur de contrôle. Ainsi, la valeur de ce paramètre est considérée comme handle du menu lorsque vous créez une fenêtre de niveau supérieur, et comme l'identificateur de contrôle lorsque vous créez une fenêtre fille.

A cet instant de l'exécution de l'application, l'identificateur de contrôle est un numéro significatif qui différencie la fenêtre fille - ou la fenêtre de contrôle - de toutes les autres fenêtres. Comme il est employé dans de nombreux messages passés entre le parent et le contrôle, et dans certaines fonctions Windows (comme CheckRadioButton), un identificateur de contrôle doit être unique.

Boîtes de dialogue et contrôles

Nous examinerons plus sérieusement les boîtes de dialogue dans un prochain article. Une petite digression s'impose néanmoins car il existe des relations inextricables entre les fenêtres de contrôle et les boîtes de dialogue.

Une boîte de dialogue est une fenêtre contenant un certain nombre de fenêtres de contrôle. Elle peut être construite en cas de besoin immédiat ou, plus généralement, définie dans un fichier ressource et chargée au moment opportun. Une boîte de dialogue peut être modale ou non modale.

Lorsqu'une boîte modale est affichée, l'utilisateur ne peut plus passer à une autre fenêtre dans le même programme. Il doit répondre d'une manière ou d'une autre à la demande de la boîte et pour ce faire, il suffit bien souvent de presser sur un bouton "OK" ou "Annuler". L'utilisateur peut cependant passer à un autre programme, sauf si la boîte ouverte est une boîte modale système, qui exige une réponse avant que l'utilisateur puisse faire quoi que ce soit d'autre dans l'environnement Windows. Lorsqu'une boîte de dialogue est

ouverte par l'utilisateur, sa fenêtre mère est désactivée et le gestionnaire de boîte de dialogue déclenche sa propre boucle interne de messages. (Ce gestionnaire est la partie du code de Windows qui crée la boîte de dialogue et qui traite ses messages. Nous dirons plus simplement "gestionnaire de dialogue".) L'utilisateur doit remplir les contrôles, et accepter ou annuler la boîte de dialogue avant que l'application puisse reprendre sa propre boucle de messages. Les boîtes de dialogue non modales n'imposent pas de telles restrictions à l'utilisateur, qui peut librement déplacer le focus hors de la boîte avant d'y répondre.

Deux des plus importantes fonctions du gestionnaire de dialogue sont le traitement des frappes de touches et l'allocation du focus d'entrée au contrôle approprié. Par exemple, lorsque vous pressez la touche de tabulation ou la touche de retour arrière, le focus devrait être respectivement déplacé vers le contrôle suivant ou précédent. Si vous pressez Enter (ou Entrée), c'est le bouton poussoir par défaut qui devrait être activé. Quand vous appuyez sur la touche Alt combinée à une autre touche, le gestionnaire de dialogue doit voir si cette combinaison correspond au raccourci (accélérateur) associé à un contrôle.

Si vous employez CreateWindow pour créer des fenêtres de contrôle au lieu de passer par le gestionnaire de dialogue, votre application doit gérer elle-même la frappe des touches du clavier et décider si telle ou telle frappe provoque un changement de focus. C'est une tâche assez ardue, qu'il est préférable de laisser au gestionnaire de dialogue.

Les contrôles statiques

Dans cet article, nous supposerons que toutes les fenêtres de contrôle (à l'exception des barres de défilement) sont associées à des boîtes de dialogue.

Un contrôle statique se distingue des autres types de contrôle à deux égards. D'abord, un contrôle statique ne peut pas recevoir le focus, et vous ne pouvez donc pas lui attribuer une entrée clavier ou une entrée souris. Ensuite, un contrôle statique n'informe pas son parent lorsque survient un événement intéressant.

Les contrôles statiques sont employés principalement pour afficher des chaînes de texte et des cadres rectangulaires. Pour avoir une idée de ce qu'est un contrôle statique typique, observez n'importe quelle boîte de dialogue contenant un contrôle d'édition : le texte du titre du contrôle constitue lui-même une fenêtre de contrôle statique. Il peut sembler curieux qu'il y ait toute une structure de fenêtre (et donc une WinProc) derrière une simple chaîne de texte, mais Windows offre un maximum de souplesse, même pour les objets les plus simples.

Un drapeau de style est associé à chaque type de contrôle statique. Pour les cadres, on peut spécifier SS_BLACKFRAME, SS_GRAYFRAME ou SS_WHITEFRAME. Pour les rectangles pleins, on peut spécifier SS_BLACKRECT, SS_GRAYRECT ou SS_WHITERECT. Le texte peut être affiché justifié à gauche, centré ou justifié à droite, à l'aide des indicateurs SS_LEFT, SS_CENTER ou SS_RIGHT. Le style SS_SIMPLE est utilisé pour afficher un simple rectangle contenant une seule ligne de texte justifié à gauche. Pour désactiver le traitement de l'esperluète (caractère &) qui associe un raccourci à un contrôle statique, il suffit de spécifier le style SS_NOPREFIX. Enfin, il est possible de dessiner ses propres contrôles statiques grâce à l'indicateur de style SS_USERITEM.

Contrôles d'édition

Windows supporte à la fois les contrôles d'entrée de données unilignes et multilignes. En fait, on peut élaborer un petit traitement de texte simplifié en créant tout bonnement un contrôle d'édition multiligne - qui est un contrôle standard de Windows, ce qui montre la puissance du logiciel. Les caractéristiques par défaut mettent à votre disposition un déplacement rudimentaire du curseur (gauche, droite, haut, bas, début et fin de ligne, début et fin de tampon, mot précédent et mot suivant) ainsi que les fonctions copier et coller. Dans Windows 3.0, les contrôles d'édition interprètent correctement les arrêts de tabulation. Bien entendu, le support de souris est complet. De plus, un contrôle d'édition multiligne exécutera automatiquement la mise à la ligne (wordwrapping).

Un contrôle d'édition multiligne possède le style ES_MULTILINE. On peut lui attacher des barres de défilement verti-

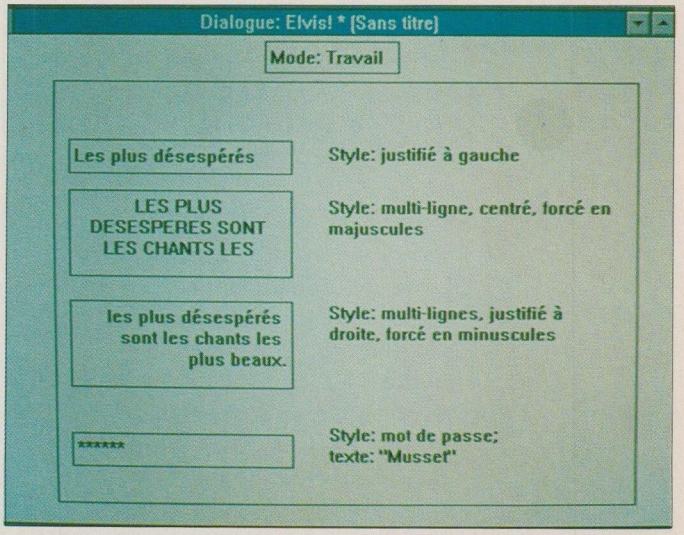


FIGURE 3 : Styles des contrôles d'édition

cales et/ou horizontales à l'aide des styles ES_AUTOVSCROLL et ES_AUTOHSCROLL. Si vous liez des barres de défilement à un contrôle d'édition, celui-ci traitera lui-même les messages de ses barres de défilement.

Avec Windows 3.0, les styles ES_LEFT, ES_CENTER et ES_RIGHT fonctionnent correctement. Ces drapeaux indiquent au contrôle d'édition que le texte contenu dans la fenêtre d'édition doit être justifié à gauche, centré, ou justifié à droite (voir Figure 3).

Windows possède encore un point faible en ce qui concerne la vérification des données. Le sous-classement des contrôles d'édition est l'un des passe-temps favoris des développeurs qui ont besoin de sous-classes de ce genre. Windows 3.0 implémente trois nouveau styles d'édition qui apportent des améliorations dans ce domaine. Les styles ES_LOWERCASE et ES_UPPERCASE transforment respectivement toutes les entrées du clavier en minuscules ou en majuscules. Le style ES_PASSWORD ordonne à un contrôle d'édition d'afficher un caractère de confidentialité (défini par l'utilisateur) à la place du texte normal. Le caractère de confidentialité par défaut est l'astérisque, mais vous pouvez le modifier à l'aide du message EM_SETPASSWORDCHAR. Espérons que les prochaines versions de Windows offriront des champs d'entrée de données formatées.

Messages d'un contrôle d'édition

Le contrôle d'édition supporte un grand nombre de messages. Certains d'entre eux demandent au contrôle d'accomplir une opération d'édition sur le texte, tandis que d'autres demandent des informations sur l'état du contrôle.

La plupart des messages du contrôle d'édition commencent par le préfixe EN_. Les quatre messages qui ne comportent pas ce préfixe concernent les opérations couper/coller. Tout texte coupé, copié ou collé dans un contrôle d'édition passe par le Presse-papiers de Windows.

Le message WM_CUT copie le texte en surbrillance dans le Presse-papiers, puis supprime ce texte dans le contrôle d'édition. Le message WM_CLEAR supprime simplement le texte sélectionné. Le message WM_PASTE insère le texte contenu dans le Presse-papiers à la position courante du curseur d'édition. Quant au message WM_UNDO, il annule la dernière opération d'édition. Un contrôle d'édition est limité à une seule annulation (on peut annuler la dernière opération, mais pas les précédentes). Dans ces messages, les paramètres wParam et lParam ne sont pas utilisés et l'on peut donc leur donner la valeur 0.

Deux des messages les plus fréquemment employés pour les fenêtres de contrôle servent à déterminer et à définir la zone de texte sélectionnée. Le message EM_GETSEL renvoie la position de départ du texte sélectionné dans le mot inférieur (LOWORD) de la valeur de retour, et la position de fin + 1 dans le mot supérieur (HIWORD). Si aucun texte n'est sélectionné, la position courante du curseur est retournée dans le mot inférieur.

```
DWORD ulRet = SendMessage(hEdit, EM_GETSEL,
0, OL);
WORD iStart = LOWORD(ulRet);
WORD iEnd = HIWORD(ulRet);
```

Le message EM_SETSEL définit la zone sélectionnée. Les positions de début et de fin de la sélection sont placées dans lParam. Si le mot inférieur de lParam est 0 et le mot supérieur 7FFFH, tout le contenu du champ d'édition est sélectionné.

```
SendMessage(hEdit, EM_SETSEL, 0, MAKELONG(0,
0x7FFF));
```

Il est possible de remplacer le texte sélectionné par un autre texte à l'aide du message EM_REPLACESEL. Dans ce message, lParam est un pointeur long sur la chaîne de texte qui sera insérée à la place de la sélection.

Si le contrôle d'édition se trouve dans une boîte de dialogue, vous pouvez employer deux fonctions pour retrouver ou définir le contenu du contrôle. Il suffit de connaître le handle de la boîte de dialogue et l'identificateur du contrôle d'édition. GetDlgItemText copie un nombre spécifié de caractères du contrôle d'édition vers un tampon défini par l'utilisateur. SetDlgItemText place une chaîne dans le contrôle d'édition. Voici une fonction qui change en majuscules le contenu d'un contrôle d'édition :

```
GetDlgItemText(hDlg, ID_EDIT, (LPSTR) szBuf,
sizeof(szBuf)-1);
strupr(szBuf);
SetDlgItemText(hDlg, ID_EDIT, szBuf);
```

Beaucoup d'autres messages d'édition peuvent être employés avec un contrôle d'édition multiligne. Pour une description complète de ces messages, chacun pourra se référer au Kit de Développement de Microsoft Windows (SDK). Les messages de notification de la Figure 4 sont générés par un contrôle d'édition lorsque certains événements se produisent; ils sont envoyés à la fenêtre mère du contrôle.

FIGURE 4
Messages de notification des contrôles d'édition

EN_CHANGE	Indique que l'utilisateur a accompli une action qui peut avoir modifié le contenu du texte.
EN_ERRSPACE	Indique qu'il n'y a plus d'espace libre dans le

.../...

```
                contrôle d'édition.
EN_HSCROLL      Indique que l'utilisateur a
                cliqué sur la barre de
                défilement horizontale du
                contrôle d'édition; la fenêtre
                mère reçoit la notification
                avant que l'écran ne soit mis
                à jour.
EN_KILLFOCUS    Indique que le contrôle
                d'édition a perdu le focus
                d'entrée.
EN_MAXTEXT      Spécifie que l'insertion
                courante a dépassé un nombre
                spécifié de caractères dans
                le contrôle d'édition.
EN_SETFOCUS     Indique que le contrôle
                d'édition a obtenu le focus
                d'entrée.
EN_UPDATE       Spécifie que le contrôle
                d'édition affichera le texte
                modifié.
EN_VSCROLL      Indique que l'utilisateur
                a cliqué sur la barre de
                défilement verticale du
                contrôle d'édition; la fenêtre
                mère reçoit la notification
                avant que l'écran ne soit mis
                à jour.
```

Contrôles boutons

"Bouton" est le terme générique désignant un contrôle qui produit un message et qui généralement change d'aspect lorsque l'utilisateur clique dans sa zone *(voir Figure 5)*. La famille des contrôles boutons comprend les "cases à cocher", les boutons radio (ou "cases d'option"), les boutons poussoir (ou "boutons de commande") et les boîtes de groupe. Une boîte de groupe ne répond pas lorsque l'on clique sur elle; elle n'est destinée qu'à contenir un groupe d'un certain autre type de boutons.

Lorsqu'on clique sur un bouton, un ou plusieurs messages WM_COMMAND sont envoyés à sa fenêtre mère. Le message WM_COMMAND indique au parent qu'un événement significatif s'est produit et concerne un bouton. Le paramètre wParam du message reçoit la valeur de l'identificateur de contrôle de la fenêtre. Le mot de poids faible (LOWORD) de lParam reçoit pour valeur le handle de contrôle de la fenêtre, et le mot de poids fort (HIWORD) reçoit le code de notification. Ce code de notification spécifie exactement quel événement s'est produit; en fonction de cette information vous pouvez annuler une boîte de contrôle ou simplement dire à Windows de modifier l'état d'un bouton.

De plus, Windows supporte les boutons owner-draw. Il s'agit de boutons "personnalisés" dessinés par le programmeur. Windows envoie un message au bouton owner-draw lorsque celui-ci change d'état et doit être réaffiché (état pressé ou relâché). Il peut être amusant de dessiner des boutons ayant l'aspect d'objets réels. Par exemple, une application musicale Windows pourrait implémenter des boutons ressemblant à ceux d'une chaîne hi-fi.

Boutons poussoir

Les boutons poussoir sont des objets rectangulaires contenant en leur centre un petit texte descriptif. Contrairement aux cases à cocher et aux boutons radio, qui sont surtout employés pour montrer l'état d'un objet, les boutons poussoir exigent une action. Quand l'utilisateur clique sur un bouton poussoir, c'est en général pour déclencher l'exécution d'une opération, par exemple l'effacement d'une boîte de dialogue ou l'activation d'une fenêtre d'aide.

Il n'existe qu'un seul style de bouton poussoir, spécifié par le drapeau BS_PUSHBUTTON. Un bouton poussoir peut posséder une propriété particulière. Si l'utilisateur presse la touche Enter (Entrée) alors qu'il se trouve dans une boîte de dialogue, le gestionnaire de dialogue déterminera s'il existe un bouton poussoir par défaut (ou "sélectionné") dans cette boîte. S'il y en a un, le gestionnaire de dialogue simulera une pression sur ce bouton; autrement dit, le gestionnaire de dialogue produira un message WM_COMMAND dont le paramètre wParam contiendra l'identificateur du bouton poussoir par défaut.

Deux messages Windows concernent les boutons poussoir

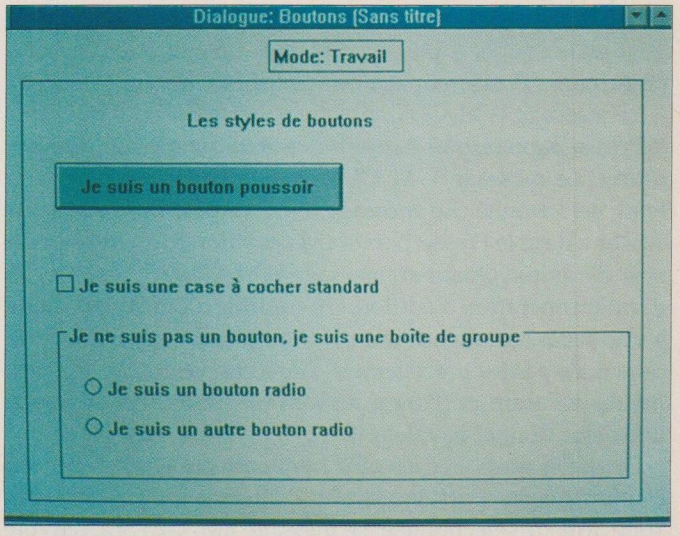


FIGURE 5 : Styles des boutons

par défaut. Le message DM_GETDEFID demande à une boîte de dialogue si elle contient un bouton poussoir par défaut. Le message DM_SETDEFID permet de sélectionner n'importe quel bouton poussoir. Le code qui suit détermine si le bouton "OK" est le bouton sélectionné d'une boîte de dialogue. Si c'est le cas, il sélectionne le bouton "Annuler".

```
  DWORD ulRet = SendMessage(hDlg, DM_GETDEFID,
0, OL);
  if (HIWORD(ulRet) == DC_HASDEFID &&
LOWORD(ulRet) == IDOK)
  SendMessage(hDlg, DM_SETDEFID,
IDCANCEL, OL);
```

Cases à cocher

Une case à cocher est un contrôle bouton et se compose de deux parties : un petit cadre rectangulaire qui peut être vide ou contenir une marque (en général un X), et une chaîne de texte décrivant le bouton, placée à droite du rectangle. Les cases à cocher caractérisent les propriétés qui peuvent être actives ou inactives. Par exemple, dans un traitement de texte, une case à cocher permet à l'utilisateur d'activer ou désactiver la fonction de mise à la ligne automatique.

Une case à cocher standard possède un des deux styles BS_CHECKBOX ou BS_AUTOCHECKBOX (style automatique). Avec ce deuxième style, Windows inverse automatiquement l'état d'une case à cocher lorsque l'utilisateur clique dans la case. Si la case ne possède pas le style automatique, Windows envoie un message WM_COMMAND au parent de la case à cocher. Il faut alors appeler CheckDlgButton pour placer ou supprimer la marque.

Une variante de la case à cocher standard est la case à cocher trois-états. En plus des états actif/inactif, une case trois-états possède un état "grisé" signifiant qu'elle est invalide (inutilisable à ce moment). Une case trois-états peut posséder l'un des deux styles BS_3STATE ou BS_AUTO3STATE. A part cela, elle est identique à la case à cocher standard.

Il existe deux fonctions API pour définir et connaître l'état d'une case à cocher. CheckDlgButton définit l'état d'une case standard ou trois-états. Voici son format:

```
  CheckDlgButton(HWND hDlg, int idButton, WORD
wCheck);
```

Le handle de la boîte de dialogue contenant la case est hDlg, et idButton est l'identificateur de contrôle de la case. wCheck a pour valeur 0 si le bouton est désactivé, 1 s'il est activé (coché), et 2 s'il est invalide (en grisé). La fonction CheckDlgButton détermine simplement le handle de fenêtre de la case à cocher et un message BM_SETCHECK à la WinProc de la case. L'état d'une case peut être connu à l'aide de la fonction IsDlgButtonChecked :

```
  IsDlgButtonChecked(HWND hDlg, int idButton);
```

Cette fonction retourne la valeur 0 si la case n'est pas cochée, 1 si elle l'est, et 2 s'il s'agit d'une case trois-états invalide.

Boutons radio

Imaginez le poste de radio d'une automobile dans les années soixante-dix, avant l'apparition des tableaux numériques. Il y avait en général un jeu de cinq boutons. Lorsque vous pressiez un bouton, tous les autres ressortaient. La classe des boutons radio de Windows fonctionne de la même manière. Les boutons radio se trouvent toujours en groupe (deux ou plus). Lorsque l'utilisateur clique sur l'un d'entre eux, celui-ci est activé et tous les autres boutons radio du groupe sont désactivés. Un bouton radio standard est défini par le style BS_RADIOBUTTON. Comme pour les cases à cocher, il existe un style BS_AUTORADIOBUTTON qui permet à Windows de gérer automatiquement l'activation des boutons.

Les fonctions CheckDlgButton et IsDlgButtonChecked définissent et testent respectivement l'état d'un bouton radio. On peut également employer la fonction CheckRadioButton pour activer un bouton du groupe.

```
  CheckRadioButon(HWND hDlg, int idFirst,
int idLast,
                  int idCheckButton);
```

Les deuxième et troisième arguments de cette fonction spécifient respectivement les identificateurs de contrôle du premier et du dernier bouton dans le groupe. Le quatrième argument fournit l'identificateur de contrôle du bouton radio que vous souhaitez activer. Pour que cette fonction puisse s'exécuter correctement, tous les boutons radio d'un même groupe doivent être numérotés consécutivement, de idFirst à idLast. CheckRadioButton passe en revue tous les contrôles dont les identificateurs se trouvent dans l'intervalle idFirst...idLast, et elle leur envoie des messages BM_SETCHECK en donnant la valeur 0 à l'argument wParam. Elle envoie ensuite un autre message BM_SETCHECK au bouton radio spécifié dans le dernier argument, mais en donnant cette fois la valeur 1 à wParam.

Boîtes de groupe

Comme nous l'avons déjà dit, les boîtes de groupe ne sont pas réellement des boutons. Il s'agit de simples rectangles

employés pour encadrer un groupe de boutons et elles ressemblent davantage aux cadres des zones de texte. Elles ne peuvent pas accepter d'entrée de données, et ne peuvent pas générer de messages WM_COMMAND; elles sont surtout utilisées pour donner un aspect esthétique à la présentation et pour séparer les groupes de boutons.

Les boîtes liste

Une boîte liste (ou "zone de liste") est une fenêtre de contrôle contenant une liste de chaînes (avec Windows 3.0, il peut s'agir d'une liste de n'importe quelle sorte d'objets affichables, mais pour l'instant nous n'examinerons que les chaînes). Pour sélectionner une ou plusieurs chaînes dans la boîte liste, l'utilisateur clique dessus ou les met en surbrillance. Si toutes les chaînes ne tiennent pas dans la fenêtre, des barres de défilement verticales ou horizontales peuvent être employées pour se déplacer dans la liste.

Windows 3.0 a considérablement accru les capacités des boîtes liste. Elles peuvent désormais être étirées horizontalement, ce qui permet des affichages sur plusieurs colonnes. Toutes sortes d'objets peuvent être affichés dans une boîte liste, et pas seulement des chaînes. Les boîtes liste peuvent maintenant posséder une tabulation pour aligner des colonnes. On peut créer une boîte liste étendue, qui permet de sélectionner plusieurs éléments grâce à des combinaisons de touches. En fait, le SDK de Windows contient un programme baptisé OWNCOMBO qui présente quelques-unes de ces nouvelles caractéristiques. Il n'est pas inutile de prendre quelques minutes pour le compiler et l'exécuter.

Les styles disponibles pour les boîtes liste sont donnés dans la Figure 6. Par défaut, une boîte liste supporte la sélection d'une seule chaîne et les chaînes ne sont pas triées. Pour les trier, il faut spécifier le style LBS_SORT. Il faut également spécifier le style LBS_NOTIFY si l'on souhaite que la boîte envoie des messages WM_COMMAND à son parent lorsqu'un événement significatif se produit. Le fichier WINDOWS.H définit le style standard LBS_STANDARD; il s'agit simplement d'une combinaison de LBS_SORT et de LBS_NOTIFY.

Maintenant que nous avons défini ce qu'est une boîte liste, il faut pouvoir y ajouter des chaînes. Pour ajouter une chaîne, l'on envoie un message à la boîte en donnant à l'argument lParam la valeur d'un pointeur long sur cette chaîne. Deux messages sont employés pour ajouter des chaînes : LB_ADDSTRING et LB_INSERTSTRING. LB_ADDSTRING ajoute une nouvelle chaîne à la fin de la liste. Si la boîte est triée, LB_ADDSTRING insère la chaîne à la position adéquate.

```
SendMessage(hListBox, LB_ADDSTRING, 0,
                (DWORD) (LPSTR)
"Hello");
```

LB_INSERTSTRING insère une chaîne à n'importe quelle position spécifiée dans la boîte. La position (d'indice initial zéro) qui est passée par wParam sera reconnue même si la boîte est triée. Si vous souhaitez placer la chaîne à la fin de la liste, il faut donner à l'argument lParam la valeur -1.

```
SendMessage(hListBox, LB_IN-
SERTSTRING, 12,
            (DWORD) (LPSTR)
"Hello");
```

Windows alloue un bloc de mémoire global de 64 Ko pour chaque boîte liste. Cela signifie que la somme des longueurs de toutes les chaînes d'une boîte liste ne doit pas excéder 64 Ko. Si vous dépassez cette limite, les messages LB_INSERTSTRING et LB_ADDSTRING renvoient à votre application la valeur LB_ERRSPACE.

Pour effacer une chaîne d'une boîte liste, utilisez le message LB_DELETESTRING et passez l'indice (base 0) de la chaîne dans le paramètre wParam.

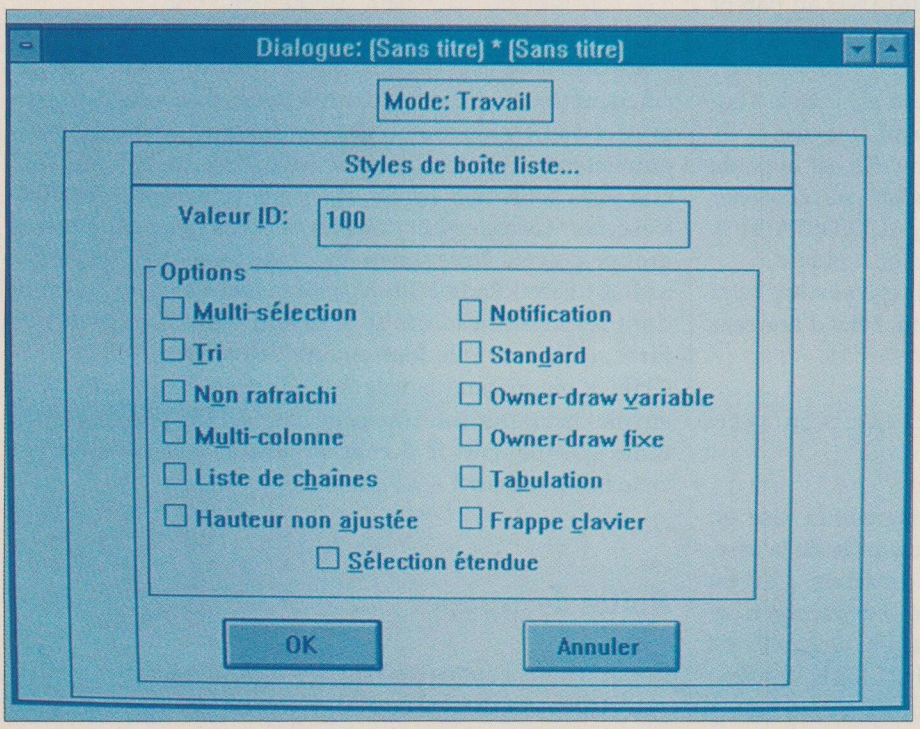


FIGURE 6 : Styles des boîtes liste

```
  SendMessage hListBox, LB_DELETESTRING, 5,
OL);
```

Malheureusement, aucun message de Windows ne permet de changer le contenu d'une chaîne. Pour cela, il faut donc supprimer la chaîne puis la réinsérer. Un phénomène déplaisant se produira si vous ajoutez ou insérez des chaînes dans une boîte liste visible. En effet, la boîte liste se rafraîchit elle-même après chaque ajout de chaîne, ce qui d'un point de vue esthétique, n'est pas très agréable pour l'utilisateur. Afin d'éviter cela, il faut dire à la boîte de ne pas se rafraîchir tant que toutes les chaînes n'ont pas été ajoutées. Le message WM_SETREDRAW active ou désactive le drapeau d'autorafraîchissement de la boîte liste. Avant de commencer à ajouter des chaînes, appelez SendMessage en donnant la valeur FALSE au troisième paramètre.

```
  SendMessage(hListBox, WM_SETREDRAW, FALSE,
0);
```

Une fois que les chaînes sont placées dans la liste et que l'utilisateur peut interagir avec la boîte, un ensemble de messages est disponible pour obtenir des informations sur l'état courant du contrôle. Avec une boîte à sélection unique, le message LB_GETCURSEL renvoie l'indice de l'élément sélectionné. Pour retrouver la chaîne associée à cet élément, employez le message LB_GETTEXT.

```
  int iSel;
  char szBuf[128];
  if ((Isel = SendMesssage(hListBox,
LB_GETCURSEL, 0,
                          OL)) !=
LB_ERR)
            SendMessage(hListBox, LB_GETTEXT,
iSel, (DWORD) (LPSTR) szBuf);
```

Pour faire passer la sélection courante à une autre chaîne, il convient d'envoyer le message LB_SESTCURSEL en donnant à l'argument wParam la valeur d'indice (de base 0) de la chaîne à sélectionner. Pour chercher une chaîne contenant un certain préfixe, il est possible d'employer le message LB_FINDSTRING. Pour accomplir ces deux opérations, on peut utiliser LB_SELECTSTRING, qui cherche une chaîne contenant un préfixe spécifié et qui la sélectionne.

Boîte liste multisélection

Dans une boîte liste multisélection, il est possible de sélectionner plusieurs chaînes en même temps. Supposons que l'on veuille présenter à l'utilisateur une liste des fichiers à effacer. Il serait fastidieux de devoir les supprimer un à un, en sélectionnant à chaque fois un fichier, puis en pressant à chaque fois la touche de suppression. Au lieu de cela, l'utilisateur peut sélectionner plusieurs fichiers dans une boîte liste multisélection, et il lui suffira ensuite de presser une seule fois la touche de suppression.

Bien que la plupart des messages des boîtes liste puissent être employés à la fois pour les boîtes à sélection unique et pour les boîtes multisélection, certains messages ne sont pas utilisables dans les deux cas. Les messages LB_GETCURSEL et LLB_SETCURSEL ne peuvent être employés qu'avec les boîtes à sélection unique, puisque plusieurs éléments peuvent être sélectionnés simultanément dans une boîte multisélection. Avec celles-ci, il faut alors utiliser les messages LB_GETSEL et LB_SETSEL afin de rechercher et de définir l'état d'un élément.

Pour travailler de manière efficace avec des messages, il sera sans doute nécessaire de savoir combien d'éléments se trouvent dans la boîte liste. Leur nombre peut être déterminé en envoyant un message LB_GETCOUNT. Une fois que vous connaissez le nombre d'éléments, il est possible d'examiner la boîte et de demander où définir l'état de chaque élément. L'exemple de code qui suit inverse l'état de sélection de tous les éléments de la boîte liste :

```
  int nItems, i;
  BOOL bisSelected;
  nItems = (WORD) SendMessage(hListBox,
LB_GETCOUNT,
  0,                        OL);
  for (i = 0; i  nItems; i++)
  {
              bisSelected = (BOOL)
SendMessage(hListBox,
LB_GETSEL, i, OL);
  SendMessage(hListBox, LB_SETSEL,
!bisSelected,
                    DWORD) i);
}
```

Deux nouveaux messages de Windows 3.0 aident à la programmation des boîtes liste multisélection. LB_GETSELCOUNT renvoie le nombre d'éléments sélectionnés dans une boîte multisélection. LB_GETSELITEMS place dans un tampon un tableau d'entiers qui fournissent les indices de tous les éléments sélectionnés. Voici un exemple qui montre comment fonctionnent ces deux messages :

```
  int nItems;
  nItems = (WORD) SendMessage(hListBox,
LB_GETSELCOUNT,
              0,
OL);
  if (nItems != LB_ERR)
```

```
  {
                HANDLE haItems =
LocalAlloc(LMEM_MOVEABLE,
nItems * sizeof(int));
                LPINT lpaItems = (LPINT)
LocalLock(haItems);
SendMessage(hListBox, LB_GETSELITEMS,
nItems,
                    DWORD) lpaTiems);
   .
   .
   .
}
```

La nouvelle police système à espacement proportionnel de Windows 3.0 peut troubler les applications Windows 2.x qui employaient l'ancienne police à espacement fixe pour aligner le contenu d'une boîte liste. Néanmoins, le nouveau support de tabulation peut être utilisé afin de placer les données en colonnes. Pour cela il faut donner à la boîte liste le style LBS_USETABSTOPS.

L'intervalle par défaut des taquets de tabulation horizontale est de 32 unités. (Une unité horizontale de boîte de dialogue est égale au quart de l'unité de largeur de la boîte de dialogue. Ce chiffre est calculé d'après la largeur de la police système courante.) Pour définir votre propre tabulation, il vous faut utiliser le message LB_SETTABSTOPS.

Boîtes liste multicolonne

Windows 3.0 propose des boîtes liste qui peuvent défiler horizontalement ou verticalement, mais pas les deux à la fois. Une boîte liste à défilement horizontal est appelée boîte multicolonne et elle est définie par le style LBS_MULTICOLUMN. Chaque colonne de la boîte liste possède une largeur par défaut de 15 caractères.

Le message LB_SETCOLUMNWIDTH permet de modifier cette largeur. La nouvelle largeur est passée dans wParam. Malheureusement, on ne peut pas avoir des colonnes à largeur variable - mais on ne peut pas tout avoir tout de suite, attendons les prochaines versions de Windows...

Comme tous les contrôles, une boîte liste informe sa fenêtre mère lorsque certains événements se produisent. Rappelons que le processus de notification n'intervient que si la boîte liste possède les styles LBS_NOTIFY ou LBS_STANDARD. Une liste des codes de notification est présentée dans la Figure 7.

FIGURE 7
Codes de notification des boîtes liste

LBN_DBLCLK	Envoyé quand l'utilisateur effectue un double-clic sur une chaîne de caractères.
LBN_ERRSPACE	Envoyé lorsque la mémoire système est pleine. Allocation mémoire impossible.
LBN_KILLFOCUS	Indique que la boîte liste a perdu le focus.
LBN_SELCHANGE	Envoyé lorsque la sélection a changé.
LBN_SETFOCUS	Indique que la boîte liste a obtenu le focus.

Les messages LBN_SETFOCUS et LBN_KILLFOCUS, introduits par Windows 3.0, informent le parent lorsqu'une boîte liste obtient ou perd le focus. Le message LBN_SETFOCUS peut être employé pour mettre en surbrillance une sélection par défaut quand l'utilisateur entre dans une boîte liste. Le message LBN_SELCHANGE est souvent utilisé lorsqu'un autre contrôle surveille le texte sélectionné dans une boîte liste. Dans Windows, par exemple, presque toutes les boîtes de dialogue relatives aux répertoires possèdent à la fois une boîte liste contenant les noms des fichiers, des répertoires et des lecteurs de disques, et une zone d'édition ("zone de texte") dans laquelle vous pouvez taper un nom de fichier. Dans la plupart des cas, si le focus est placé sur la boîte liste, le texte de l'élément sélectionné est affiché "simultanément" dans la zone d'édition. Pour cela, il faut savoir à quel moment la sélection change. On obtient cette information grâce au message LBN_SELCHANGE.

A chaque fois que la sélection change dans une boîte liste, un message WM_COMMAND est envoyé à la boîte de dialogue, avec un code de notification LBN_SELCHANGE dans le mot de poids fort de lParam. On peut alors récupérer le texte de l'élément sélectionné, en utilisant le code que nous avons présenté un peu plus haut, et ce texte est placé dans la zone d'édition à l'aide de la fonction SetDlgItemText.

Le message LBN_BBLCLK est généré quand l'utilisateur clique deux fois sur un élément d'une boîte liste. Un double-clic signifie généralement que l'utilisateur souhaite sélectionner une chaîne et effacer la boîte de dialogue.

Boîtes liste owner-draw

Dans le précédent article, nous avons examiné les éléments de menu owner-draw, pour lesquels Windows laisse à l'ap-

plication en cours toute la responsabilité de l'affichage. Ce concept s'applique également aux boîtes liste : l'utilisateur peut choisir de placer n'importe quoi dans une boîte liste, et pas seulement des chaînes de caractères. Cela permet de créer des interfaces utilisateur plus efficaces puisque l'on peut placer notamment un dessin d'un objet au lieu d'un texte descriptif.

Si notre application d'exemple (le portefeuille d'actions) supportait différents types de graphes, elle pourrait présenter à l'utilisateur des dessins de ces différents types pour qu'il puisse faire son choix. Une représentation miniature d'un graphe incliné serait plus compréhensible que les mots "Graphe incliné".

Pour créer une boîte liste owner-draw, vous devez donner à celle-ci les styles LBS_OWNERDRAWFIXED ou LBS_OWNERDRAWVARIABLE.

Une boîte liste créée avec LBS_OWNERDRAWFIXED suppose que tous les éléments auront la même hauteur; en revanche, une boîte créée avec LBS_OWNERDRAWVARIABLE doit connaître la hauteur de chaque élément.

Si l'on souhaite remplir une boîte avec des chaînes affichées dans des polices différentes (par exemple pour demander à l'utilisateur de choisir une police), il faut donner à cette boîte le style LBS_HASSTRING en plus de l'un des deux styles mentionnés précédemment.

Supposons que nous avons créé une boîte liste owner-draw et que nous souhaitons y ajouter des éléments. Nous pouvons utiliser les fonctions normales LB_ADDSTRING ou LB_INSERTSTRING. Cependant, l'argument lParam n'a pas besoin d'être un pointeur long sur une chaîne, sauf si la boîte possède le style LBS_HASSTRING.

L'argument peut être une valeur sur 32 bits définie par l'utilisateur : par exemple un handle d'image bitmap, un indice de tableau ou toute autre valeur utilisable pour associer une boîte liste à un objet graphique.

Considérons notre application de portefeuille d'actions. Il est souhaitable d'offrir à l'utilisateur un choix de grilles pour afficher les lignes d'un graphe. Nous avons créé dans ce but des grilles avec différents stylos (ou "crayons") de Windows. (Un stylo ou crayon est une image bitmap employée pour tracer une ligne.) Windows possède un stylo à motif plein et plusieurs stylos à motif hachuré.

Une boîte liste owner-draw correspond très bien au choix que nous voulons proposer. Au lieu de décrire les différents types de lignes par des chaînes de caractères, nous pouvons créer une liste dont les éléments seront des lignes dessinées avec des stylos différents.

La première chose à faire est de créer une boîte liste ayant le style LBS_OWNERDRAWFIXED, car toutes les lignes auront la même taille.

Il faut ensuite ajouter les éléments. Chacun d'eux est associé à une valeur définie par l'utilisateur : il s'agira ici des valeurs prédéfinies par Windows pour chaque style de stylo.

```
/* Remplir la liste avec les ID des stylos */
                int iPen;

for (iPen = 0; iPen  PS_NULL; iPen++)
  SendMessage(hPenListBox, LB_ADDSTRING, 0,
(LONG) iPen);
```

Comment faire si la boîte liste owner-draw doit être triée (autrement dit, si elle possède le style LBS_SORT) ? La boîte ne sait pas comment trier des objets graphiques. Heureusement, Windows 3.0 propose un nouveau message, WM_COMPAREITEM, qui est envoyé à la fenêtre mère de la boîte liste à chaque fois que deux objets de la liste doivent être triés. Quand ce message est envoyé, la valeur de lParam est un pointeur long sur une structure de données de type COMPAREITEMSTRUCT. Parmi les éléments de cette structure se trouvent les indices et les valeurs utilisateur des deux éléments à comparer. Trois valeurs de retour sont possibles : 0 si les objets sont égaux, -1 si l'élément 1 doit précéder l'élément 2, et 1 si l'élément 1 doit être placé après l'élément 2.

Windows a besoin de connaître la taille de chaque élément d'une boîte liste owner-draw. Pour obtenir cette information, le message WM_MEASUREITEM est envoyé à la fenêtre mère de la boîte liste. Si la boîte possède le style LBS_OWNERDRAWFIXED, ce message n'est envoyé qu'une seule fois car cet indicateur signifie que tous les éléments ont la même taille. Dans un message WM_MEASUREITEM, la valeur de l'argument lParam est un pointeur long vers une structure MEASUREITEMSTRUCT. Cette structure a été présentée dans l'article précédent et nous n'y reviendrons pas.

Supposons que tous les éléments de la boîte de sélection de lignes ont une largeur de 42 pixels et une hauteur de 20 pixels. Le code qui suit traite le message WM_MEASUREITEM:

```
case WM_MEASUREITEM :
{
  LPMEASUREITEM STRUCT lpm1 =
              (LPMEASUREITEMSTRUCT) (LPSTR)
lParam;
  lpMI -itemWidth =
42;
  lpMI - itemHeight = 20;
  break;
}
```

Enfin, pour dessiner chaque élément, Windows envoie au parent de la boîte liste le message WM_DRAWITEM. Nous avons emprunté un petit extrait du programme de démonstration OWNCOMBO SDK pour montrer comment traiter ce message *(voir figure 8)*. Il faut pouvoir dessiner un élément

de la boîte liste dans son état normal (non sélectionné). Bien entendu, il faut pouvoir également dessiner un élément dans son état sélectionné, et le présenter d'une manière spécifique lorsqu'il a le focus - pour le cas où il se trouve dans une boîte liste multisélection. (Quand une chaîne a le focus dans une boîte liste standard, Windows entoure le texte d'un cadre en pointillé.)

FIGURE 8
Le message WM_DRAWITEM

```
case WM_DRAWITEM :
  {
    LPDRAWITEMSTRUCT lpdis = (LPDRAWITEM-
STRUCT) (LPSTR) lParam;
    switch (lpdis-itemAction)
    {
      case ODA_DRAWENTIRE:
        DrawEntireItem(lpdis, -4);
        break;
      case ODA_SELECT:
        HandleSelectionState(lpdis, 0);
        break;
      case ODA_FOCUS:
        HandleFocusState(lpdis, -2);
        break;
    }
    break;
  }

/*FONCTION : HandleSelectionState(LPDRAWI-
TEMSTRUCT, int) *
  DESCRIPTION: Gère un changement de
  sélection d'un élément. Si un élément est
  sélectionné, il est entouré d'un cadre
  noir (surbrillance); si un élément est
  dé-sélectionné, le cadre disparaît.
  COMMENTAIRE: Le cadre de sélection noir
  est un peu plus grand que le cadre de
  focus gris, et ils ne se superposeront
  pas.*/

void FAR PASCAL HandleSelection-
State(lpdis, inflate)
  LPDRAWITEMSTRUCT lpdis;
  int              inflate;
{
  RECT rc;
  HPBRUSH hbr;
/*  Redéfinit le rectangle pour placer la
  sélection en dehors du focus du cadre et
  de l'élément.
*/
                                    .../...
  CopyRect ((LPRECT)&rc, (LPRECT)&lpdis-rcI
tem);
  InflateRect ((LPRECT) &rc, inflate, in-
flate;
  if (lpdis-itemState & ODS_SELECTED)
  {
/*  sélectionner l'élément - afficher un
  cadre noir */
  hbr = GetStockObject(BLACK_BRUSH);
  }
  else
  {
/*  désélectionner - effacer le cadre */
  hbr = CreateSolidBrush(GetSysColor(CO-
LOR_WINDOW));
  }
  FrameRect(lpdis-hDC, (LPRECT) &rc, hbr);
  DeleteObject (hbr);
}

/*FONCTION : HandleFocusState(LPDRAWITEM-
STRUCT, int)
  DESCRIPTION: Gère un changement de focus.
  Si un objet obtient le focus, il est
  entouré d'un cadre rectangulaire gris;
  s'il perd le focus le cadre est supprimé.
  COMMENTAIRE: Le cadre de focus est un peu
  plus petit que le cadre de sélection noir,
  pour qu'ils ne se confondent pas.*/

void FAR PASCAL HandleFocusState(lpdis, in-
flate)
  LPDRAWITEMSTRUCT lpdis;
  int              inflate;
{
  RECT rc;
  HBRUSH hbr;
  /* Redéfinit la taille du rectangle */
  CopyRect ((LPRECT)&rc, (LPRECT)&lpdis-rcI-
tem);
  InflateRect ((LPRECT)&rc, inflate, in-
flate);
  if (lpdis-itemState & ODS_FOCUS)
    /* obtient le focus - dessine un cadre
gris */
    hbr = GetStockObject(GRAY_BRUSH);
  else
    /* perd le focus - efface le cadre */
    hbr = CreateSolidBrush(GetSysColor(CO-
LOR_WINDOW));
  FrameRect(lpdis-hDC, (LPRECT)&rc, hbr);
  DeleteObject (hbr);
}
                                    .../...
```

```
/*  FONCTION : DrawEntireItem(LPDRAWITEM-
STRUCT, int)
  DESCRIPTION: Dessine un objet et l'entoure
  d'un cadre de sélection ou d'un cadre de
  focus.*/

void FAR PASCAL DrawEntireItem(lpdis, in-
flate)
  LPDRAWITEMSTRUCT lpdis;
  int              inflate;
{
  RECT   rc;
  HANDLE hOldPen;
  HPEN   hPen;
  /*Redéfinit la taille du rectangle afin de
laisser de la place pour les cadres */
  CopyRect ((LPRECT)&rc, (LPRECT)&lpdis-rcI-
tem);
  InflateRect ((LPRECT)&rc, inflate, in-
flate);
  hPen = CreatePen((int) lpdis-itemData, 1,
RGB(0, 0, 0));
  hOldPen = SelectObject(lpdis-hDC, hPen);
  MoveTo(lpdis-hDC, rc.left, rc.top +
(rc.bottom - rc.top) / 2);
  LineTo(lpdis-hDC, rc.right, rc.top +
(rc.bottom - rc.top) / 2);
  SelectObject(lpdis-hDC, hOldPen);
  DeleteObject(hPen);
  /* Affiche ou efface le cadre approprié */
  HandleSelectionState(lpdis, inflate + 4);
  HandleFocusState(lpdis, inflate + 2);
}
```

Lorsqu'un élément est supprimé d'une boîte liste owner-draw, Windows envoie à la fenêtre mère un message WM_DELETEITEM. Dans ce message, l'argument lParam pointe vers une structure DELETEITEMSTRUCT, qui contient l'indice et la valeur utilisateur correspondant à cet élément. WM_DELETEITEM donne ainsi à l'utilisateur une possibilité de libérer des ressources associées à l'élément.

Le portefeuille d'actions

Dans notre application d'exemple, les contrôles commenceront à jouer leur rôle après que les boîtes de dialogue auront été intégrées dans la nouvelle mouture du programme *(voir Langages & Systèmes n°3)*. Mais comme nous désirons ajouter dès maintenant des fonctionnalités à l'application, nous l'avons "recouverte" d'une couche MDI ("Multiple Document Interface"). Chaque graphique d'une action peut en effet être considéré comme un document MDI indépendant, et l'application, considérée dans son ensemble, s'adapte parfaitement à la structure d'interface utilisateur supportée par les spécifications MDI *(voir Figure 9)*.

FIGURE 9
Fichiers de l'application d'exemple

```
#    Define the dependencies
stock.obj : stock.c
     cl -c -Gsw -Od -W2 -Zdpi -AS stock.c
stock.res : stock.rc
     rc -r stock.rc
stock.exe: stock.obj stock.def stock.res
     d:\msc6\link /nod /co stock, stock.exe,
/map/li, slibcew libw, stock
     rc stock.res

#include "windows.h"
#include "stock.h"
ID_STOCK ICON stock.ico
ID_GRAPH ICON graph.ico
StockAccelerators ACCELERATORS
BEGIN
       "^A", ID_TICK_ADD
       "^C", ID_TICK_CHANGE
       "^O", ID_OPEN
       "^P", ID_PRINT
       "^S", ID_SAVE
END
StockMenu MENU
BEGIN
   POPUP "&Fichier"
   BEGIN
     MENUITEM"&Nouveau...",ID_NEW
     MENUITEM"&Ouvrir...\tCTRL+O",ID_OPEN
     MENUITEM"&Sauvegarder...\tCTRL+S"
ID_SAVE
     MENUITEM"&Fermer",ID_CLOSE
     MENUITEM"Im&primer\tCTRL+P",ID_PRINT
     MENUITEM"&Quitter",ID_EXIT
     MENUITEMSEPARATOR
     MENUITEM"&Mode d'affichage...",ID_MAP
   END
   POPUP "&Edition"
   BEGIN
     MENUITEM "&Ajouter une cota-
tion...\tCTRL+A",
                          ID_TICK_ADD
     MENUITEM "&Changer une cota-
tion...\tCTRL+C",
                          ID_TICK_CHANGE
   END
                                        .../...
```

```
POPUP "&Graphe"
   BEGIN
     MENUITEM "&Prix", ID_GRAPH_PRICE
     MENUITEM "&Volume", ID_GRAPH_VOLUME
     MENUITEM "&Zoom", ID_GRAPH_ZOOM
     MENUITEM "&Options...", ID_GRAPH_OPTIONS
     MENUITEM "&Couleurs...",ID_GRAPH_COLORS
     POPUP"&Grille"
     BEGIN
       MENUITEM "&Horizontale",ID_GRAPH_GRID_
HORZ
       MENUITEM "&Verticale", ID_GRAPH_GRID_
VERT
     END
   END
   POPUP "&Fenêtre"
   BEGIN
     MENUITEM "&Mosaïque", ID_WINDOW_TILE
     MENUITEM "&Cascade", ID_WINDOW_CASCADE
     MENUITEM "Réorganiser les &Icônes",
                          ID_WINDOW_ICONS
     MENUITEM "&Tout fermer",ID_WINDOW_
CLOSEALL
   END
   POPUP "\a&F1=Aide"
   BEGIN
     MENUITEM"&A propos de Stock...",ID_ABOUT
   END
END
ID_ABOUT DIALOG LOADONCALL MOVEABLE DISCAR-
DABLE 25, 23, 184, 78
   CAPTION "A propos de MSJ Stock"
   STYLE WS_BORDER | DS_MODALFRAME | WS_
   CAPTION | WS_DLGFRAME |
         WS_POPUP| WS_SYSMENU
BEGIN
   CONTROL "MSJ Stock" -1,"static", SS_CENTER |
WS_GROUP | WS_CHILD, 0, 15, 184, 8
   CONTROL "OK" IDOK, "button", BS_DEFPUSH-
     BUTTON | WS_GROUP | WS_TABSTOP
     | WS_CHILD,76, 60, 32, 14
   CONTROL ID_STOCK, -1, "static", SS_ICON
     | WS_CHILD, 25, 14, 16, 21
END
MAPPING_MODES DIALOG LOADONCALL MOVEABLE
   DISCARDABLE 67, 31, 242, 126
CAPTION "Modes d'affichage"
STYLE WS_BORDER | WS_CAPTION | WS_DLGFRAME
   | WS_SYSMENU | WS_POPUP
BEGIN
     CONTROL "Fenêtre Org X", -1, "static",
       SS_LEFT | WS_CHILD, 2, 9, 57, 8
                                        .../...
```

```
     CONTROL "", ID_WXORG, "edit", ES_LEFT |
       WS_BORDER | WS_TABSTOP | WS_CHILD |
       ES_AUTOHSCROLL, 60, 7, 17, 12
     CONTROL "Fenêtre X Ext", -1, "static",
       SS_LEFT | WS_CHILD, 1, 24, 54, 8
     CONTROL "", ID_WYORG, "edit", ES_LEFT |
       WS_BORDER | WS_TABSTOP | WS_CHILD |
       ES_AUTOHSCROLL, 91, 7, 13, 12
     CONTROL "Y", -1, "static", SS_LEFT |
       WS_CHILD, 195, 8, 7, 11
     CONTROL "", ID_WXEXT, "edit", ES_LEFT |
       WS_BORDER | WS_TABSTOP | WS_CHILD |
       ES_AUTOHSCROLL, 58, 23, 23, 12
     CONTROL "Fenêtre Y Ext", -1, "static",
       SS_LEFT | WS_CHILD, 2, 40, 55, 8
     CONTROL "", ID_WYEXT, "edit", ES_LEFT |
       WS_BORDER | WS_TABSTOP | WS_CHILD |
       ES_AUTOHSCROLL, 58, 39, 25, 12
     CONTROL "Zone Org X", -1, "static",
       SS_LEFT | WS_CHILD, 112, 9, 58, 9
     CONTROL "", ID_VXORG, "edit", ES_LEFT |
       WS_BORDER | WS_TABSTOP | WS_CHILD |
       ES_AUTOHSCROLL, 173, 8, 19, 12
     CONTROL "Y", -1, "static", SS_LEFT |
       WS_CHILD, 80, 9, 11, 8
     CONTROL "", ID_VYORG, "edit", ES_LEFT |
       WS_BORDER | WS_TABSTOP | WS_CHILD |
       ES_AUTOHSCROLL, 205, 7, 14, 12
     CONTROL "Zone X Ext", -1, "static",
       SS_LEFT | WS_CHILD, 113, 25, 55, 10
     CONTROL "", ID_VXEXT, "edit", ES_LEFT |
       WS_BORDER | WS_TABSTOP | WS_CHILD |
       ES_AUTOHSCROLL, 170, 23, 30, 12
     CONTROL "Zone Y Ext", -1, "static",
       SS_LEFT | WS_CHILD, 113, 41, 55, 9
     CONTROL "", ID_VYEXT, "edit", ES_LEFT |
       WS_BORDER | WS_TABSTOP | WS_CHILD |
       ES_AUTOHSCROLL, 171, 39, 29, 12
     CONTROL "OK", 1, "button", BS_DEFPUSH-
       BUTTON | WS_TABSTOP | WS_CHILD, 30,
       67,24, 14
     CONTROL "Annuler", 2, "button",
       BS_PUSHBUTTON | WS_TABSTOP | WS_CHILD
       129, 67, 30, 14
     CONTROL "Rect. client:", ID_RECTTEXT,
       "static", SS_LEFT | WS_CHILD, 9, 97,
       226, 11
END
ADDTICK DIALOG LOADONCALL MOVEABLE DISCAR-
DABLE 112, 31, 106, 86
   CAPTION "Ajouter une cotation"
   STYLE WS_BORDER | WS_CAPTION | WS_DLGFRAME
     | WS_POPUP
                                        .../...
```

```
BEGIN
     CONTROL "Date:", -1, "static", SS_LEFT |
       WS_CHILD, 2, 7, 22, 8
     CONTROL "", ID_TICK_DATE, "edit",
       ES_LEFT | WS_BORDER | WS_TABSTOP |
       WS_CHILD, 53, 5, 48, 12
     CONTROL "Prix de fermeture:", -1,
     "static", SS_LEFT | WS_CHILD, 2, 26,
     55, 11
     CONTROL "", ID_TICK_PRICE, "edit",
       ES_LEFT | WS_BORDER | WS_TABSTOP |
       WS_CHILD, 57, 25, 44, 12
     CONTROL "Volume:", -1, "static", SS_LEFT |
       WS_CHILD, 2, 44, 32, 8
     CONTROL "", ID_TICK_VOLUME, "edit",
       ES_LEFT | WS_BORDER | WS_TABSTOP |
       WS_CHILD, 38, 43, 63, 12
     CONTROL "OK", 1, "button", BS_DEFPUSH-
       BUTTON | WS_TABSTOP | WS_CHILD, 9, 66,
       28, 14
     CONTROL "Annuler", 2, "button", BS_PUSH-
       BUTTON | WS_TABSTOP | WS_CHILD, 63,
       66, 32, 14
END
Open DIALOG 10, 10, 148, 112
STYLE DS_MODALFRAME | WS_CAPTION |
   WS_SYSMENU
CAPTION "Ouvrir "
BEGIN
       LTEXT "&Nom du fichier:", IDC_FILENAME,
        4,4,60, 10
       EDITTEXT                  IDC_EDIT,
        4, 16, 100, 12, ES_AUTOHSCROLL
       LTEXT "&Fichiers dans",   IDC_FILES
        4, 40,32, 10
       LISTBOX,                  IDC_LISTBOX,
        4, 52,70, 56, WS_TABSTOP
       LTEXT "",                 IDC_PATH,
        40, 40, 100, 10
       DEFPUSHBUTTON "&Ouvrir",  IDOK,
        87, 60,50, 14
       PUSHBUTTON "Annuler",     IDCANCEL,
        87, 80,50, 14
END
GRAPH DIALOG LOADONCALL MOVEABLE DISCAR-
   DABLE 48, 32, 212, 125
CAPTION "Information"
STYLE WS_BORDER | WS_CAPTION | WS_DLGFRAME
   | WS_SYSMENU | WS_POPUP
BEGIN
     CONTROL "Symbole de l'action:", -1,
       "static", SS_LEFT | WS_CHILD,
       4, 7, 49, 8
                                        .../...
```

```
     CONTROL " ", ID_SYMBOL, "edit", ES_LEFT |
        WS_BORDER | WS_TABSTOP | WS_CHILD,
       60, 7, 32, 12
     CONTROL "Prix Minimum:", -1, "static",
       SS_LEFT | WS_CHILD, 4, 29, 54, 8
     CONTROL "", ID_MINPRICE, "edit",
       ES_LEFT | WS_BORDER | WS_TABSTOP |
       WS_CHILD, 60, 27, 32, 12
     CONTROL "Prix Maximum:", -1, "static",
       SS_LEFT | WS_CHILD, 4, 48, 57, 10
     CONTROL "", ID_MAXPRICE, "edit",
       ES_LEFT | WS_BORDER | WS_TABSTOP |
       WS_CHILD, 60, 47, 32, 12
     CONTROL "Facteur d'échelle:", -1,
       "static", SS_LEFT | WS_CHILD,
       4, 68, 46, 8
     CONTROL "", ID_FACTOR, "edit", ES_LEFT |
       WS_BORDER | WS_TABSTOP | WS_CHILD,
       60, 67, 32, 12
     CONTROL "Intervalle de cotation:", -1,
       "static", SS_LEFT | WS_CHILD,
       4, 88, 44, 8
     CONTROL "", ID_TICKINT, "edit",
       ES_LEFT | WS_BORDER | WS_TABSTOP |
       WS_CHILD, 60, 87, 32, 12
     CONTROL "Unité:", -1, "static",
       SS_LEFT | WS_CHILD, 4, 108, 65, 8
     CONTROL "", ID_DENOMINATOR, "edit",
       ES_LEFT | WS_BORDER | WS_TABSTOP |
       WS_CHILD, 69, 107, 18, 12
     CONTROL "Grilles", -1, "button",
       BS_GROUPBOX | WS_TABSTOP | WS_CHILD,
       100, 8, 55, 109
     CONTROL "Verticale", ID_VERTGRID,
       "button", BS_AUTOCHECKBOX | WS_TABSTOP
       | WS_CHILD, 103, 19, 37, 12
     CONTROL "Horizontale", ID_HORZGRID,
       "button", BS_AUTOCHECKBOX | WS_TABSTOP
       | WS_CHILD, 103, 32, 45, 12
     CONTROL "Style de grille", -1, "static",
       SS_LEFT | WS_CHILD, 103, 46, 36, 8
     CONTROL "", ID_GRIDSTYLE, "combobox",
       CBS_DROPDOWNLIST | CBS_OWNERDRAWFIXED
       | WS_VSCROLL | WS_CHILD, 104, 56, 46,
       58
     CONTROL "OK", 1, "button", BS_DEFPUSH-
       BUTTON | WS_TABSTOP | WS_CHILD,
       170, 12, 29, 14
     CONTROL "Annuler", 2, "button", BS_PUSH-
       BUTTON | WS_TABSTOP | WS_CHILD,
       170, 38, 29, 14
     CONTROL "Aide", ID_HELP, "button",
                                        .../...
```

```
        BS_PUSHBUTTON | WS_TABSTOP | WS_CHILD,
        170, 63, 29, 14
END
STRINGTABLE
BEGIN
   ID_NEW,"Créer une nouvelle base de données"
   ID_OPEN,"Ouvrir un fichier existant"
   ID_SAVE,"Sauvegarder le fichier courant"
   ID_CLOSE,"Fermer le fichier courant"
   ID_PRINT,"Imprimer le graphique courant"
   ID_TICK_ADD,"Ajouter une cotation"
   ID_TICK_CHANGE,"Modifier les cotations"
   ID_GRAPH_PRICE,"Afficher un graphe du prix"
   ID_GRAPH_VOLUME,"Afficher un graphe
     du volume"
   ID_GRAPH_ZOOM,"Zoom sur le graphe actif"
   ID_GRAPH_COLORS,"Changer les couleurs
     du graphe"
   ID_GRAPH_GRID_HORZ,"Ajouter/Supprimer la
     grille horizontale"
   ID_GRAPH_GRID_VERT,"Ajouter/Supprimer la
     grille verticale"
   ID_ABOUT,"Afficher une information
     concernant l'application"
   ID_EXIT,"Quitter l'application"
   1,   "Nouveau, Ouvrir, Fermer, Imprimer,
        Quitter"
   2,   "Ajouter et modifier les cotations"
   3,   "Affichage des graphiques"
   4,   "Réorganiser les fenêtres"
   5,   "Aide sur le programme MSJ Stock"
END


/* Fichier : STOCK.H
   Objet :
   Historique :
   (C) Copyright 1990 Marc Adler/Magma
     Systems
   Tous droits réservés */

extern HANDLE hThisInstance; /* Handle
   d'instance programme */
extern HWND hWndMain;        /* Handle de
   la  fenêtre principale */
extern HWND hwndMDIClient;   /* Handle du
   client MDI */
extern HWND hWndActive;
extern HWND hWndStatus;
   #define Y_STATUSWINDOW 20 /* Hauteur de la
     fenêtre d'état */
extern HANDLE hCurrStockInfo;
                                        .../...
```

```
#define CBWNDEXTRA          sizeof(HANDLE)
/*
Information à propos de l'action
*/
#define MAXSTOCKNAME     4
#define MAXDESCRIPTION   32
typedef struct tagDate
{
   BYTE                  chMonth;
   BYTE                  chDay;
   BYTE                  chYear;
} DATE;
typedef struct tagPrice
{
   DWORD                 dwTotal;
   /* montant en dollar * iDenominator
   + fraction */
} PRICE;
typedef struct tagTick
{
   PRICE                 price;
   DWORD                 dwVolume;
   DATE date;
} TICK, FAR *LPTICK;
/*
   Structure décrivant l'affichage du graphe
*/
typedef struct tagGraphInfo
{
   DWORD dwMinPrice;
   DWORD dwMaxPrice;
   DWORD dwScaleFactor;
   DWORD dwTickInterval;
   WORD  iDenominator;
   WORD  iGridPen;
} GRAPHINFO, FAR *LPGRAPHINFO;
typedef struct tagStockFile {
   DWORD dwMagic;
#define MAGIC_COOKIE  66666666L
   char szStock[MAXSTOCKNAME];
   char szDescription[MAXDESCRIPTION];
   GRAPHINFO graphinfo;
   WORD nTicks;
/*
   TICK aTicks[1];
*/
} STOCKFILE;
typedef struct tagInCoreStockInfo
{
   char     szFileName[14];
            /* fichier contenant les données
               de l'action */
   STOCKFILE StockFile;                 .../...
```

```
            /* copie de l'en-tête du fichier
               de l'action */
   HANDLE hTicks;
   HWND     hWnd;
            /* fenêtre d'affichage de
               l'action */
   DWORD    dwFlags;
            /* toutes sortes de bits d'état
               nécessaires */
#define STATE_HAS_VGRID 1L
#define STATE_HAS_HGRID 2L
HANDLE hNextStockInfo;
/* lien vers l'action suivante */
} STOCKINFO, FAR *LPSTOCKINFO;

#define ID_ABOUT                    99
#define ID_STOCK                   100
#define ID_GRAPH                   101

#define ID_NEW                     101
#define ID_OPEN                    102
#define ID_SAVE                    103
#define ID_CLOSE                   104
#define ID_PRINT                   105
#define ID_EXIT                    106
#define ID_MAP                     107

#define ID_TICK_ADD                200
#define ID_TICK_CHANGE             201
#define ID_GRAPH_PRICE             202
#define ID_GRAPH_VOLUME            203
#define ID_GRAPH_ZOOM              204
#define ID_GRAPH_COLORS            205
#define ID_GRAPH_OPTIONS           199
#define ID_GRAPH_GRID_HORZ         206
#define ID_GRAPH_GRID_VERT         207
#define ID_WINDOW_TILE             208
#define ID_WINDOW_CASCADE          209
#define ID_WINDOW_ICONS            210
#define ID_WINDOW_CLOSEALL         211
#define ID_WINDOW_CHILDREN         220

#define ID_TICK_DATE               100
#define ID_TICK_PRICE              101
#define ID_TICK_VOLUME             102

#define ID_RECTTEXT                120
#define ID_VXEXT                   109
#define ID_VXORG                   107
#define ID_VYEXT                   111
#define ID_VYORG                   113
#define ID_WXEXT                   103
```

.../...

```
#define ID_WXORG                   101
#define ID_WYEXT                   105
#define ID_WYORG                   115

#define IDC_FILENAME               400
#define IDC_EDIT                   401
#define IDC_FILES                  402
#define IDC_PATH                   403
#define IDC_LISTBOX                404

/*
   Eléments pour la boîte de dialogue
   du graphe
*/
#define ID_SYMBOL                  300
#define ID_MINPRICE                301
#define ID_MAXPRICE                302
#define ID_FACTOR                  303
#define ID_TICKINT                 304
#define ID_HORZGRID                305
#define ID_VERTGRID                306
#define ID_GRIDSTYLE               307
#define ID_HELP                    308
#define ID_DENOMINATOR             309

int NEAR PASCAL WinMain(HANDLE, HANDLE,
     LPSTR, int);
BOOL FAR PASCAL InitializeApplica-
     tion(VOID);
BOOL FAR PASCAL InitializeInstance(LPSTR
     lpCmdLine, WORD nCmdShow);
LONG FAR PASCAL MainWndProc(HWND, WORD,
     WORD, LONG);
LONG FAR PASCAL StatusWndProc(HWND, WORD,
     WORD, LONG);
LONG FAR PASCAL GraphWndProc(HWND, WORD,
     WORD, LONG);
BOOL FAR PASCAL AboutDlgProc(HWND, WORD,
     WORD, LONG);
BOOL FAR PASCAL MapDlgProc(HWND, WORD,
     WORD, LONG);
BOOL FAR PASCAL AddTickDlgProc(HWND, WORD,
     WORD, LONG);
BOOL FAR PASCAL GraphInfoDlgProc(HWND,
     WORD, WORD, LONG);
VOID MainPaint(HWND, HDC);
int PASCAL StockFileRead(LPSTR szFileName);
HWND PASCAL GraphCreateWindow(LPSTR
     lpName);
int PASCAL GraphWndPaint(HWND hWnd, HDC
     hDC,LPSTOCKINFO lpStockInfo);
/* FILEOPEN.C */
```

.../...

```
HANDLE FAR PASCAL OpenDlg(HWND, unsigned,
     WORD, LONG);
void SeparateFile(HWND, LPSTR, LPSTR,
     LPSTR);
void UpdateListBox(HWND);
void AddExt(PSTR, PSTR);
void ChangeDefExt(PSTR, PSTR);


NAME                    STOCK
DESCRIPTION             'MSJ Stock
                        Application'
EXETYPE                 WINDOWS
STUB                    'WINSTUB.EXE'
CODE                    FIXED
DATA                    MOVEABLE MULTIPLE
HEAPSIZE                2048
STACKSIZE               8192
EXPORTS MainWndProc
        StatusWndProc
        GraphWndProc


/*
Microsoft Systems Journal Stock Application

     Par Marc Adler
     Magma Systems
     15 Bodwell Terrace
     Millburn, New Jersey07041

Ce code accompagne la troisième partie de
la série d'articles consacrée à l'appren-
tissage de Windows. C'est l'ossature de no-
tre application. Nous ajoutons maintenant
le support MDI.
*/
#include <stdio.h>
#include <string.h>
#include "windows.h"
#include "stock.h"
/*
   Variables globales
*/
HANDLE hThisInstance;/* Handle: instance
   du programme */
HWND hWndMain= NULL       /* Handle: fenêtre
   principale*/
HWND hwndMDIClient = NULL; /* Handle du
   client MDI */
HWND hWndActive= NULL; /* Handle of the
   active MDI child */
```

.../...

```
HWND hWndStatus= NULL; /* Fenêtre des
   messages d'état */
HANDLE hAccelTable;/* Handle: table des
   raccourcis*/
int NEAR PASCAL WinMain(hInstance,
   hPrevInstance,
lpszCmdLine, nCmdShow)
   HANDLE hInstance;
   HANDLE hPrevInstance;
   LPSTR  lpszCmdLine;
   int    nCmdShow;
{
   MSG msg;
   /*
     Sauvegarder le handle de cette instance
   */
   hThisInstance = hInstance;
   /*
     S'il s'agit de la première instance de
     l'application recenser les classes de
     fenêtre...
   */
   if (!hPrevInstance)
   {
     if (!InitializeApplication())
     {
       MessageBeep(0);
       return FALSE;
     }
   }
   /*
     Procéder à l'initialisation de cette
     instance particulière...
   */
   if (!InitializeInstance(lpszCmdLine,
       nCmdShow))
     return FALSE;
   /*
     La boucle de messages principale...
   */
   while (GetMessage(&msg, (HWND) NULL, 0,
     0))
   {
   /* Si un message clavier est destiné au
     MDI, laisser le client MDI s'en occuper
     Sinon, vérifier s'il s'agit d'un
     raccourci normal (comme F3 = Poursuite
     de la recherche). Si ce n'est pas le
     cas traiter le message selon la méthode
     ordinaire.
   */
   if  (!TranslateMDISysAccel(hwndMDIClient,
```

.../...

```
    &msg) &&
      !TranslateAccelerator(hWndMain,
    hAccelTable, &msg))
    {
      TranslateMessage(&msg);
      DispatchMessage(&msg);
    }
  }
  return 0;
}
BOOL FAR PASCAL InitializeApplication(VOID)
{
  WNDCLASS wc;
  /*
    Recenser la classe de la fenêtre
    principale
  */
  wc.style= CS_VREDRAW | CS_HREDRAW;
  wc.lpfnWndProc = MainWndProc;
  wc.cbClsExtra= 0;
  wc.cbWndExtra= 0;
  wc.hInstance= hThisInstance;
  wc.hIcon= LoadIcon(hThisInstance,
       MAKEINTRESOURCE(ID_STOCK));
  wc.hCursor= LoadCursor(NULL, IDC_ARROW);
  wc.hbrBackground = GetStockOb-
       ject(WHITE_BRUSH);
  wc.lpszMenuName= "StockMenu";
  wc.lpszClassName = "StockMainWindow";
  if (!RegisterClass(&wc))
    return FALSE;
  /*Enregistrer la classe de la fenêtre
    fille MDI */
  wc.lpfnWndProc = GraphWndProc;
  wc.hIcon= LoadIcon(hThisInstance,
       MAKEINTRESOURCE(ID_GRAPH));
  wc.lpszMenuName= NULL;
  wc.cbWndExtra= CBWNDEXTRA;
  wc.lpszClassName = "GraphWindow";
  if (!RegisterClass(&wc))
  {
    return FALSE;
  }
  wc.style= CS_VREDRAW | CS_HREDRAW;
  wc.lpfnWndProc = StatusWndProc;
  wc.cbClsExtra= 0;
  wc.cbWndExtra= 0;
  wc.hInstance= hThisInstance;
  wc.hbrBackground = GetStockOb-
    ject(WHITE_BRUSH);
  wc.lpszMenuName= NULL;
  wc.lpszClassName = "StockStatus";
```

.../...

```
  if (!RegisterClass(&wc))
    return FALSE;
  return TRUE;
}
BOOL FAR PASCAL InitializeInstance(LPSTR
  lpCmdLine, WORD nCmdShow)
{
  /*
    Créer la fenêtre principale
  */
  hWndMain = CreateWindow(
       "StockMainWindow",
       "MSJ Stock Application",
       WS_OVERLAPPEDWINDOW | WS_CLIPCHILDREN,
       CW_USEDEFAULT, 0,
       CW_USEDEFAULT, 0,
       (HWND) NULL,
       (HMENU) NULL,
       hThisInstance,
       (LPSTR) NULL);
  if (!hWndMain)
  {
    return FALSE;
  }
  /*Le client MDI devrait être créé en
    réponse à la fonction WM_CREATE pour
    la fenêtre principale.
  */
  if (!hwndMDIClient)
    return FALSE;
  /*
    Charger les raccourcis
  */
  hAccelTable = LoadAccelerators(hThisIn
  stance,(LPSTR) "StockAccelerators");
  /*
    Afficher la fenêtre principale
  */
  ShowWindow(hWndMain, nCmdShow);
  UpdateWindow(hWndMain);
  return TRUE;
}
LONG FAR PASCAL MainWndProc(hWnd, msg,
  wParam, lParam)
  HWND hWnd;
  WORD msg;
  WORD wParam;
  LONG lParam;
{
  BOOL                  rc;
  HDC                   hDC;
  PAINTSTRUCT           ps;
```

.../...

```
   FARPROC                lpfn;
   LPSTOCKINFO            lpStockInfo;
   char                   szMsg[80];
   RECT                   rClient;
   int                    i;
   static HWND ahwndSubMenus[16];
   switch (msg)
   {
     case WM_CREATE:
   {
       CLIENTCREATESTRUCT ccs;
       /* Chercher le menu qui affichera les
          fenêtres filles */
       ccs.hWindowMenu= GetSubMenu
          GetMenu(hWnd), 3);
       ccs.idFirstChild = ID_WINDOW_CHILDREN;
       /* Créer le client MDI en remplissant
          la zone cliente */
       hwndMDIClient = CreateWindow
           ("mdiclient",

                          NULL,
                          WS_CHILD | WS_CLIP-
CHILDREN,
WS_VSCROLL |WS_HSCROLL,
                          0,0,0,0,
                          hWnd,
                          0xCAC,
                          hThisInstance,
                          (LPSTR) &ccs);
       ShowWindow(hwndMDIClient, SW_SHOW);
       /*
          Créer une fenêtre d'état au bas
          de la fenêtre principale.
       */
       GetClientRect(hWnd, (LPRECT) &rc);
       hWndStatus = CreateWindow
         ("StockStatus",
                    NULL,
                    WS_CHILD | WS_BORDER,
                    rClient.left, rClient.
bottom - Y_STATUSWINDOW,
                    rClient.right - rClient.
left, Y_STATUSWINDOW,
                    hWnd,
                    NULL,
                    hThisInstance,
                    (LPSTR) 0L);
       if (hWndStatus)
        ShowWindow(hWndStatus, SW_SHOW);
       /*
        Nous rassemblons les handles de
```

.../...

```
        fenêtre de tous les sous-menus.
       */
       for (i=0;
            (ahwndSubMenus[i]=GetSubMenu
GetMenu(hWnd),i));
            i++)
         ;
       break;
     }
     case WM_COMMAND :
       switch (wParam)
       {
         case ID_NEW                :
           GraphCreateWindow(NULL);
           break;
         case ID_OPEN               :
           break;
         case ID_SAVE               :
           break;
         case ID_CLOSE              :
           break;
         case ID_PRINT              :
           break;
         case ID_EXIT               :
           PostQuitMessage(0);
           break;
         case ID_TICK_ADD           :
           break;
         case ID_TICK_CHANGE        :
           break;
         case ID_GRAPH_OPTIONS      :
           break;
         case ID_GRAPH_PRICE        :
           break;
         case ID_GRAPH_VOLUME       :
           break;
         case ID_GRAPH_ZOOM         :
           break;
         case ID_GRAPH_COLORS       :
           break;
         case ID_GRAPH_GRID_HORZ    :
           break;
         case ID_GRAPH_GRID_VERT    :
           break;
         case ID_WINDOW_TILE        :
           SendMessage(hwndMDIClient,
WM_MDITILE, 0, 0L);
           break;
         case ID_WINDOW_CASCADE     :
           SendMessage(hwndMDIClient,
WM_MDICASCADE, 0, 0L);
           break;
```

.../...

```
        case ID_WINDOW_ICONS      :
          SendMessage(hwndMDIClient,
WM_MDIICONARRANGE, 0, 0L);
          break;
        case ID_WINDOW_CLOSEALL   :
        {
          HWND hwndT;
          ShowWindow(hwndMDIClient,
SW_HIDE);
          while ((hwndT = GetWindow
(hwndMDIClient, GW_CHILD)) != NULL)
          {
            /* Laisser les fenêtres de
titre d'icône */
            while (hwndT && GetWindow
(hwndT, GW_OWNER))
              hwndT = GetWindow(hwndT,
GW_HWNDNEXT);
            if (!hwndT)
              break;
            SendMessage(hwndMDIClient,
WM_MDIDESTROY, (WORD) hwndT, 0L);
          }
          ShowWindow(hwndMDIClient, SW_SHOW);
          ShowWindow(hWndStatus, SW_SHOW);
          break;
        }
        case ID_ABOUT :
          break;
        default :
          /*
            Nous aurions pu choisir de
            passer le focus à une des
            fenêtres filles MDI.
          */
          return DefFrameProc(hWnd,
hwndMDIClient, msg, wParam, lParam);
      }
      break;
    case WM_MENUSELECT:
    {
      WORD idMenu;
      WORD iLen;
      /*
       Nous nous chargeons du message
       WM_MENUSELECT afin de pouvoir
       afficher des messages d'information
       pour chaque option du menu.
      */
      if (lParam & MF_POPUP)
      {
      /*
```

.../...

```
        Le handle du menu pop-up est renvoyé
        dans wParam. Examiner le tableau des
        handles des sous-menus afin de savoir
        dans quel sous-menu nous nous
        trouvons.
       */
        for (idMenu = 0;
            ahwndSubMenus[idMenu] && wParam !=
                        ahwndSubMenus[idMenu];
            idMenu++)
          ;
        if (!ahwndSubMenus[idMenu])
          break;
        idMenu++;/* valeur avec décalage
de 1 */
       }
       else
        /*
        Un identificateur d'élément de menu
        est renvoyé dans wParam
     * /
        idMenu = wParam;
       /*
        Si lParam était 1,0, l'utilisateur
        a abandonné le menu et nous voulons
        seulement effacer le texte d'aide
        affiché dans la fenêtre d'état.
        Sinon, idMenu contient un
        identificateur valide de menu ou de
        sous-menu. Charger la chaîne
        correspondante à partir du tableau de
        chaîne, et afficher cette chaîne.
       */
       if (LOWORD(lParam) == -1 && HIWORD
(lParam) == 0||
          ((iLen = LoadString(hThisInstance,
idMenu,
                              (LPSTR) szMsg,
sizeof(szMsg)))))
       {
        HDC hDC = GetDC(hWndStatus);
        HBRUSH hBrush, hOldBrush;
        /*
        Utiliser un pinceau blanc pour
        l'effacement.
        */
        hBrush = CreateSolidBrush(0x00FFFFFF);
        hOldBrush = SelectObject(hDC, hBrush);
        /*
        Effacer le texte d'aide précédent,
        'il existe. Puis afficher le nouveau
        message d'aide.
```

.../...

```
        */
        GetClientRect(hWndStatus, (LPRECT)
&rc);
        FillRect(hDC, &rc, hBrush);
        if (LOWORD(lParam) == -1 && HIWORD
(lParam) == 0)
           ;
        else
           TextOut(hDC, 0, 0, szMsg, iLen);
        /*
        Restaurer l'ancien pinceau.
        */
        SelectObject(hDC, hOldBrush);
        DeleteObject(hBrush);
        ReleaseDC(hWndStatus, hDC);
      }
      break;
    }
    case WM_SIZE :
      /*
       Quand la taille de la fenêtre
       principale est modifiée, il faut
       déplacer et redéfinir la taille de
       la fenêtre d'état pour qu'elle soit
       toujours placée en bas.
      */
      if (hWndStatus)
      {
       GetClientRect(hWnd, (LPRECT)
&rClient);
       MoveWindow(hWndStatus,
                  rClient.left, rClient.
bottom - Y_STATUSWINDOW,
                  rClient.right - rClient.
left, Y_STATUSWINDOW,
                  TRUE);
      }
      /*
       Il faut appeler DefFrameProc,
       car elle traite également WM_SIZE!
      */
      goto call_DFP;
    case WM_DESTROY :
      PostQuitMessage(0);
      break;
    default :
       /* Utiliser DefFrameProc() au lieu
       de DefWindowProc() car le support
       MDI demande un traitement particulier
       */
  call_DFP;
    return DefFrameProc(hWnd, hWndMDIClient,
                                        .../...
```

```
msg, wParam, lParam);
  }
  return (LONG) TRUE;
}
LONG FAR PASCAL StatusWndProc(hWnd, msg,
wParam, lParam)
  HWND hWnd;
  WORD msg;
  WORD wParam;
  LONG lParam;
{
  return DefWindowProc(hWnd, msg, wParam,
lParam);
}

/* Fonction : GraphCreateWindow()*/
  Description : Crée une fenêtre fille MDI
affichant le graphe d'une action.
  Retour: Le handle de la fenêtre fille MDI
venant d'être créée. */

HWND PASCAL GraphCreateWindow(LPSTR lpName)
{
  HWND hWnd;
  char sz[160];
  MDICREATESTRUCT mcs;
  staticintnChildren = 1;
  if(!lpName)
  {
       /*
       Si le paramètre lpName est NULLL,
       charger la chaîne "Sans titre" du
       tableau STRINGTABLE et installer le
       champ de titre de la structure MDI
       CreateStruct.
       */
    sprintf(sz, "(Sans titre - %d)", nChil-
dren++);
    mcs.szTitle = (LPSTR) sz;
  }
  else
  {
       /* Donner le titre spécifié à la
       fenêtre */
    AnsiUpper(lpName);
    mcs.szTitle = lpName;
  }
  mcs.szClass = "GraphWindow";
  mcs.hOwner = hThisInstance;
  /* Attribuer la taille par défaut à
    la fenêtre */
  mcs.x = mcs.cx = CW_DEFAULT;
                                        .../...
```

```
   mcs.y = mcs.cy = CW_DEFAULT;
   /* Attribuer au style DWORD de la fenêtre
     la valeur par défaut */
   mcs.style = OL;
   /* Demander au client MDI de créer la
       fenêtre fille */
   hWnd = (WORD) SendMessage(hwndMDIClient,
                 WM_MDICREATE,
                 0,
                 (LONG)(LPMDICREATESTRUCT)
&mcs);return hWnd;
}

/* Fonction : GraphWndProc()
   Description : Procédure de fenêtre des
   fenêtres filles MDI.
   Retour:*/

LONG FAR PASCAL GraphWndProc(hWnd, msg,
wParam, lParam)
   HWND hWnd;
   WORD msg;
   WORD wParam;
   LONG lParam;
{
   switch (msg)
   {
     case WM_MDIACTIVATE:
       /* Si nous activons cette fenêtre
         fille, il faudra s'en souvenir */
       hWndActive = (wParam) ? hWnd : (HWND)
NULL;
       hCurrStockInfo = GetWindowWord(hWnd,
0);
       break;
     default:
       /* Précisons encore que le
        comportement MDI par défaut est un
        peu différent. Il faut appeler
        DefMDIChildProc au lieu de
        DefWindowProc()
       */
       return DefMDIChildProc(hWnd, msg,
wParam, lParam);
   }
   return (LONG) FALSE;
}
```

La rubrique "Nouveau" du menu "Fichier" a été associée à la routine de création de la fenêtre fille MDI. Quand cette rubrique est sélectionnée, une nouvelle fenêtre fille MDI est créée et placée à une position par défaut dans la zone cliente. Pour l'instant, rien n'est encore affiché dans les fenêtres filles; c'est au fil du développement de l'application que ces fenêtres seront remplies avec les graphiques des actions.

Le menu déroulant "Fenêtre" vous permet de choisir le type de positionnement des filles MDI sur l'écran *(voir Figure 10)*. Vous avez le choix entre l'affichage automatique en cascade ou en mosaïque. Chaque fenêtre peut être réduite en icone; et les icones sont réorganisées automatiquement dans le bas de la fenêtre cliente MDI. Il est possible de fermer toutes les fenêtres filles en même temps grâce à la rubrique "Fenêtre/Tout fermer".

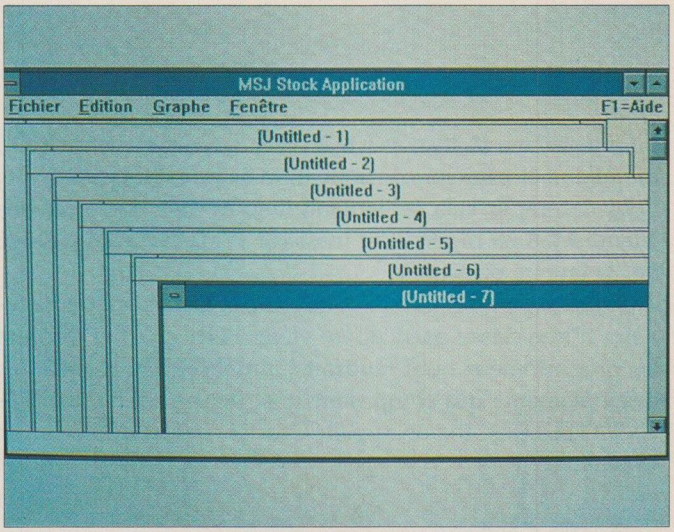


FIGURE 10 : Fenêtres MDI en cascade

Une partie du code MDI a été empruntée sans vergogne au programme de démonstration MULTIPAD, fourni avec le SDK de Windows. Le SDK propose nombre de bons exemples et de techniques de programmation. Il est vivement conseillé de s'y reporter si l'on ne connaît pas encore très bien Windows, ou si l'on désire examiner quelques-unes des nouvelles caractéristiques de Windows 3.0.

Dans la nouvelle version de notre application, la première modification concerne le code de la boucle principale, qui doit maintenant reconnaître les raccourcis MDI. Ainsi, avant d'appeler la fonction TranslateAccelerator, il faut appeler TranslateMDISysAccel. Si une de ces fonctions retourne une valeur différente de zéro, cela signifie que la frappe a été convertie.

```
  if (!TranslateMDISysAccel(hwndMDIClient,
&msg) &&
      !TranslateAccelerator(hwndMain,
hAccelTable,
                            &msg))
```

La deuxième modification est l'enregistrement d'une classe de fenêtres pour les fenêtres filles MDI. Rien dans le

code ne révèle que les fenêtres filles seront des fenêtres MDI.

```
/* Enregistrer la classe des fenêtres filles
MDI */
  wc.lpfnWndProc = GraphWndProc;
  wc.hIcon = LoadIcon(hThisInstance,
MAKEINTRESOURCE(ID_GRAPH));
  wc.lpszMenuName = NULL;
  wc.cbWndExtra = CBWNDEXTRA;
  wc.lpszClassName = "GraphWindow";
  if (!RegisterClass(&wc))
  {
              return FALSE;
}
```

La troisième tâche consiste à créer la fenêtre cliente MDI, ce qui doit être fait lors de la création de la fenêtre principale. Pour accomplir cela facilement, il suffit de créer la fenêtre cliente MDI en réponse au message WM_CREATE envoyé par la fenêtre principale.

Le message WM_CREATE informe la WinProc qu'une fenêtre d'une classe particulière vient d'être créée et indique au programmeur qu'il faudrait l'initialiser. On remplit les deux éléments qui composent la structure MDI CLIENTCREATESTRUCT et on appelle CreateWindow afin de créer le client MDI.

```
case WM_CREATE:
{
              CLIENTCREATESTRUCT ccs;
  /* Chercher le menu donnant la liste des
     fenêtres filles */
              ccs.hWindowMenu =
GetSubMenu(GetMenu(hWnd), 3);
              ccs.idFirstChild =
ID_WINDOW_CHILDREN;
  /* Créer le client MDI en remplissant la
     zone cliente */
  hwndMDIClient = CreateWindow("mdiclient",
                        NULL,
                        WS_CHILD |
                        WS_CLIPCHILDREN |
                        WS_VSCROLL |
WS_HSCROLL,
                        0, 0, 0, 0,
                        hWnd,
                        0xCAC,
                        hThisInstance,
                        (LPSTR) &ccs);
ShowWindow(hwndMDIClient, SW_SHOW);
```

Il faut encore ajouter du code à la WinProc de la fenêtre principale pour traiter quelques commandes relatives à la MDI. Ce code demande au client MDI de s'afficher en cascade ou en mosaïque, de fermer les fenêtres filles MDI et de réorganiser les icones.

Il faut bien noter que la WinProc par défaut d'une fenêtre contenant un client MDI n'est pas DefWindowProc. Avec une fenêtre de ce genre, il faut appeler DefFrameProc pour gérer tous les messages par défaut.

```
  default :
              /*
              Nous aurions pu choisir
              d'attribuer le focus à l'une
              des fenêtres filles MDI.
              */
              return DefFrameProc(hWnd,
hwndMDIClient, msg,
                        wParam,
lParam);
```

La dernière modification exigée par le support MDI est la création d'une fenêtre fille MDI lorsque l'utilisateur sélectionne la rubrique "Fichier/Nouveau". Le message WM_COMMAND est traité dans la WinProc de la fenêtre principale; si wParam a la valeur de l'identificateur ID_NEW, il faut appeler GraphCreateWindow.

GraphCreateWindow se contente de remplir une structure MDICREATESTRUCT et d'envoyer un message WM_MDICREATE à la fenêtre cliente MDI *(voir Figure 11)*. Cette structure de données MDICREATESTRUCT contient des informations sur la classe de la fenêtre fille, son titre, ses dimensions, son handle d'instance et son style.

FIGURE 11
Structure MDICREATESTRUCT

```
MDICREATESTRUCT mcs;
static int nChildren = 1;
if (!lpName)
{
   /* Si le paramètre lpName est NULLL,
   charger la chaîne Sans titre" du tableau
   STRINGTABLE et installer le champ de titre
   de la structure MDI CreateStruct.
   */
   sprintf(sz, "(Sans titre - %d)", nChil-
dren++);
   mcs.szTitle = (LPSTR) sz;
}
else
{
/* Donner le titre spécifié à la fenêtre */
   AnsiUpper(lpName);
   mcs.szTitle = lpName;
}
```

.../...

```
mcs.szClass = "GraphWindow";
mcs.hOwner = hThisInstance;
/* Attribuer la taille par défaut à la
   fenêtre */
mcs.x = mcs.cx = CW_DEFAULT;
mcs.y = mcs.cy = CW_DEFAULT;
/* Attribuer une valeur par défaut au
   style DWORD de la fenêtre */
mcs.style = OL;
/* Demander au client MDI de créer la
   fenêtre fille */
hWnd = (WORD) SendMessage(hwndMDIClient,
                  WM_MDICREATE,
                  0,
                  (LONG) (LPMDICREATESTRUCT)
                    &mcs);
return hWnd;
```

La WinProc d'une fenêtre fille MDI n'est pas associée à un code de traitement spécial, mais il faut se souvenir d'appeler DefMDIChildProc au lieu de DefWindowProc lorsqu'on ne souhaite pas qu'elle traite un message.

```
   defaut :
   /* Précision, comme le comportement MDI par
   défaut est un peu différent, appeler
   DefMDIChildProc au lieu de DefWindowProc()
   */
   return DefMDIChildProc(hWnd, msg,
wParam, lParam);
```

Et c'est tout ! Il suffit donc de quelques instructions pour ajouter le support MDI à notre application. Dans les versions précédentes de Windows, MDI était extrêmement compliqué, mais Windows 3.0 a rendu son emploi beaucoup plus facile.

Le prochain article terminera notre panorama des fenêtres de contrôle, avec l'étude des boîtes combo et des barres de défilement. Le thème principal sera l'utilisation des boîtes de dialogue, avec un bref regard sur quelques outils déjà commercialisés qui peuvent aider le programmeur à créer des boîtes de dialogue. Il restera ensuite à intégrer ces boîtes à notre application. □

Article extrait du MSJ nº 6 du mois de novembre 1990 et traduit par Henry-Luc Planchat.

Service lecteurs nº 5 (dernière page)

DEVELOP

HIGH SCREEN 5

Générateur d'écrans

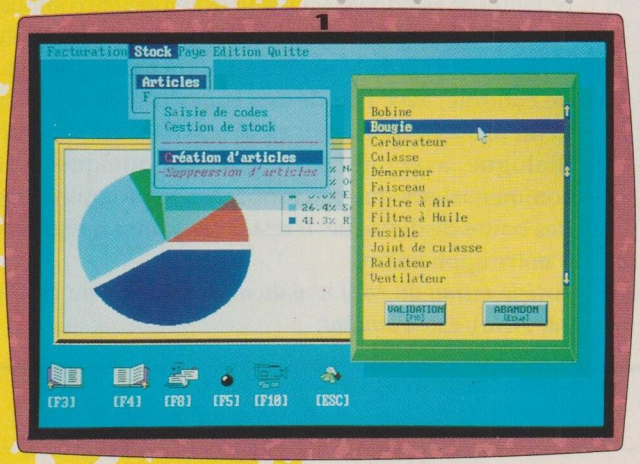
- DOS, WINDOWS, OS/2, UNIX.
- Tous langages (oui, tous!)

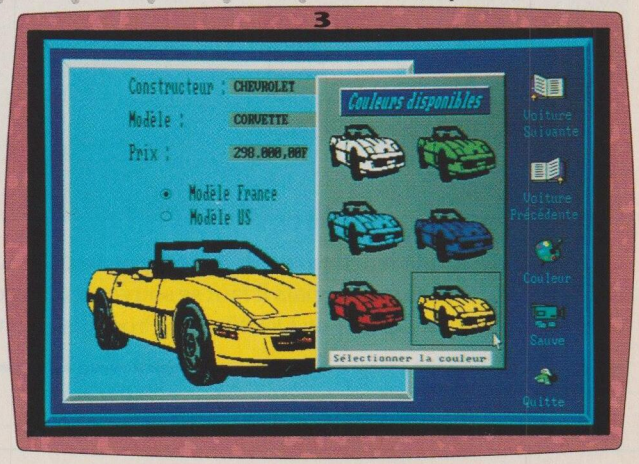
1- Menus déroulants, fenêtres, saisies, icônes, graphiques,... : tout High Screen 5 en un écran !

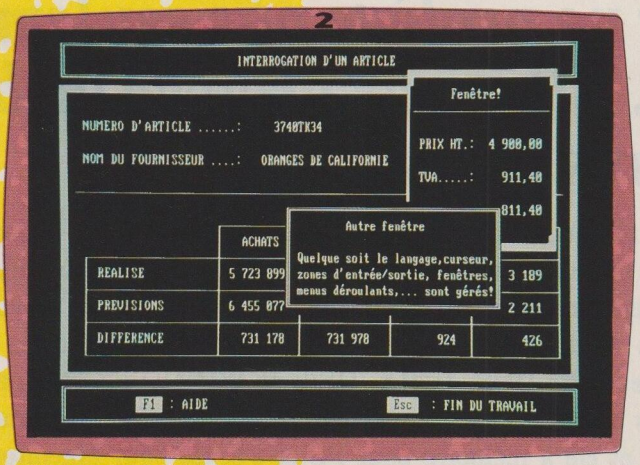
2- High Screen fonctionne sur tous les écrans...

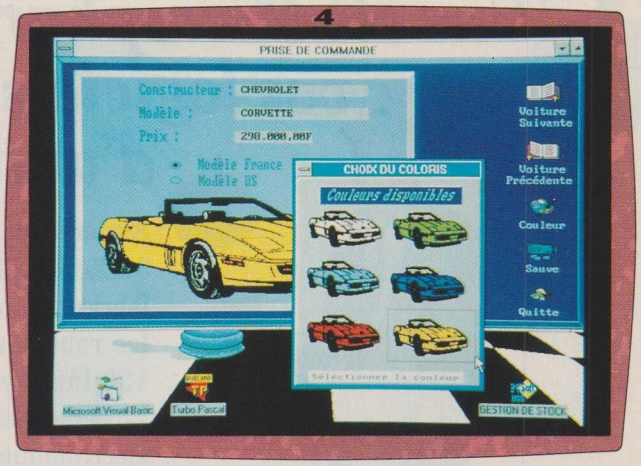
3- Une application graphique sous DOS...

4-... la même sous Windows (compatibilité !)









Que vos applications soient en mode texte ou graphique, au clavier ou à la souris, avec menus déroulants, avec des icones, avec ou sans fenêtres,
High Screen 5 vous permettra de développer 3 à 10 fois plus vite.

La programmation est très simple, avec des ordres en français.
De nombreux utilitaires sont livrés : éditeur d'icones, de dossiers, maquettage,... Tout devient possible, dans tous les langages!
(C, Basic, Pascal, dBase, Nantucket, Cobol, Fortran,...)

High Screen 5 a déjà changé la vie de milliers de développeurs : essayez-le !

A partir de
4900 F HT
5 811,40 F TTC

Disquette de démonstration disponible : appelez !

Nos produits ont reçu de nombreuses récompenses :

HYPER PACK — UTILS DE DEVELOPPEMENT — PRIX D'EXCELLENCE — micro systèmes

Meilleur outil de développement — SOFT & MICRO

LABO QUALIFIE — INFO PC

LE SUPPORT TECHNIQUE
Un des points forts de PC SOFT est le support technique, efficace, disponible et... sympathique.
Notre équipe de techniciens chevronnés est à votre disposition. Gratuitement.

GARANTIE "ESSAI SANS RISQUE"
Si le produit ne vous convenait pas pour une raison quelconque, vous pourriez nous le retourner (en état d'origine bien sûr) dans les 10 jours suivant sa réception pour un remboursement intégral. Conditions au dos du tarif.

Malgré le soin apporté à la rédaction de ce document, les informations ne sont pas contractuelles. Toutes les marques sont déposées par leur propriétaire respectif.

PEZ VITE

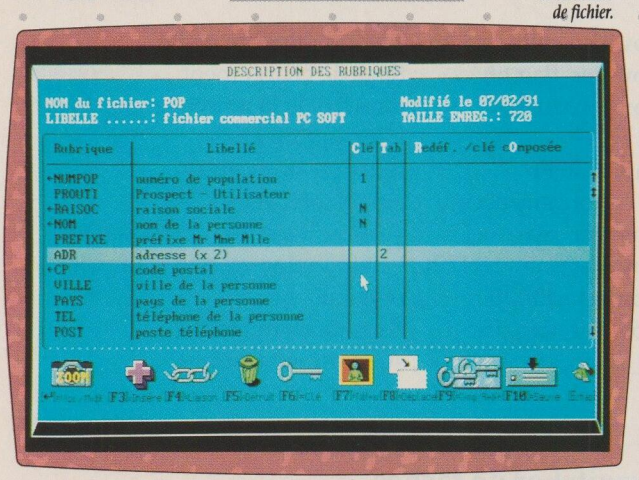
Salon "SOFTWARE DEVELOPMENT" Stand D20

HYPER FILE 3

SGBD Gestionnaire de fichiers

- DOS - WINDOWS - OS/2
- C - Basic - Pascal

Création de structure de fichier.



Hyper File 3 est le plus rapide des SGBD. Après avoir défini facilement la structure des fichiers, leur utilisation dans les programmes est très simple.
La programmation (en C, Basic, Pascal) s'effectue en français.
De plus Hyper File 3 édite automatiquement le dossier d'analyse.
De nombreux utilitaires sont livrés.

Si vous manipulez des fichiers, testez Hyper File !

A partir de
4900 F HT
5 811,40 F TTC

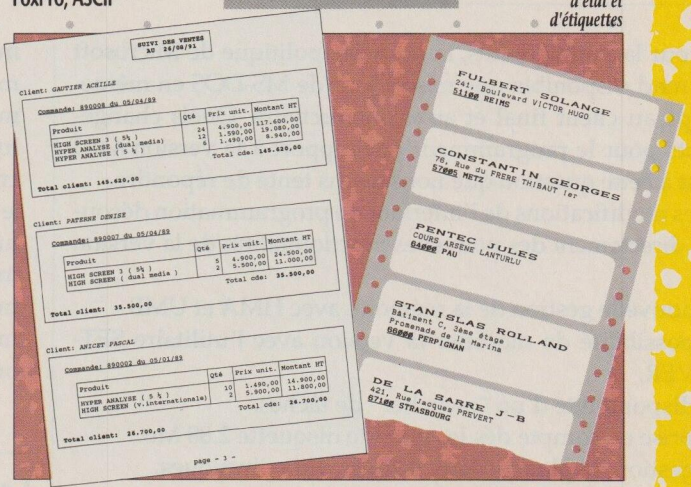
Disquette de démonstration disponible : appelez !

HYPER PRINT 3

Générateurs d'états et d'interrogations

- C, Basic, Pascal, dBase, Nantucket, FoxPro, ASCII

Exemple d'état et d'étiquettes



Réaliser des états imprimés de gestion en quelques minutes seulement, c'est possible avec Hyper Print 3.
Hyper Print 3 rapproche jusqu'à 9 fichiers par état et sait calculer. Le résultat s'édite sur imprimante (matricielle ou laser) ou à l'écran, ou même dans une feuille de tableur.

Un produit à évaluer d'urgence !

A partir de
4900 F HT
5 811,40 F TTC

Disquette de démonstration disponible : appelez !

HYPER PACK DEVELOPPEUR

Atelier de Génie Logiciel

C - Basic - Pascal

Ce "pack" regroupe High Screen, Hyper File et Hyper Print qui utilisent le même dictionnaire de données.
Le "PACK" permet de développer de superbes applications rapides, très très vite !

A partir de
9900 F HT
11741,40 F TTC

PC SOFT est Fournisseur Officiel de la Préparation Olympique

PCSOFT

LIVRAISON RAPIDE
En France métropolitaine, livraison en 24 heures par transporteur.

DISQUETTE DE DEMONSTRATION DISPONIBLE

DOCUMENTATIONS CLAIRES EN FRANCAIS

PAS DE REDEVANCES

SIEGE MONTPELLIER
Tél. (16) 67.032.032
216, rue des Escarceliers, BP 3019
34034 Montpellier Cedex 01
Fax: (16) 67.03.07.87
Support Technique: (16) 67.03.17.17

PARIS
Tél. (1) 48.01.48.88
34, Boulevard Haussmann
75009 Paris
Télex: 290.266 F (MBI)
Minitel: 3614 code PCSOFT

Service lecteurs n° 6 (dernière page)

MS-DOS 5.0

Une version essentielle

JEAN-FRANÇOIS DUVIVIER

Pour la première fois, la nouvelle politique de Microsoft rend disponible une mise à jour de MS-DOS en même-temps au client final et aux constructeurs. Quels changements pour le programmeur par rapport à la version 4.0 ? C'est à cette question que nous avons tenté de répondre.

Les modifications de l'interface de programmation découlent directement des nouvelles fonctionnalités de la version 5.0 :

- nouvelle gestion de la mémoire avec HMA et UMB
- possibilité de modifier la version avec l'utilitaire SETVER.
- disponibilité d'un "switcher" de tâches.
- prise en compte des lecteurs de disquette 2.88 Mo.
- gestion de plus de deux disques durs physiques.
- auto-documentation par la commande /?
- incorporation de certaines fonctions du SHARE dans le noyau.

Cependant, un immense pas a été aussi réalisé par la documentation, officialisant ainsi des fonctions longtemps considérées comme "cachées" mais que tout programme résident un peu complexe ne pouvait se permettre de ne pas utiliser *(cf encadré TSR)*.

La version du DOS

La disponibilité de l'utilitaire SETVER rend soudainement les choses plus compliquées. SETVER permet en effet à un utilisateur de manipuler à sa guise le numéro de version du DOS qui sera renvoyé à un programme. Si cela peut solutionner quelques problèmes avec des programmes trop regardants sur le numéro de version, il y a fort à parier que SETVER en créera beaucoup plus en faisant croire à un vieil utilitaire qu'il tourne toujours sous 2.0 !! Un programme devra donc dorénavant suivre la procédure suivante :

```
mov  ax,3001h          ; numéro de l'ancienne fonction.
int  21h               ; interruption DOS.
mov  ver_maj,al        ; version majeure.
mov  ver_maj_reel,al   ; version majeure réelle.
mov  ver_min,ah        ; version mineure.
mov  ver_min_reel,ah   ; version mineure réelle.
mov  ver_flag,bh       ; flags de chargement.
mov  ax,3306h          ; numéro de la nouvelle fonction.
int  21h               ; interruption DOS.
cmp  al,0FFh           ; fonction supportée ?
je   non_supporte      ; non si inférieure à 5.0.
mov  ver_maj_reel,bl   ; version majeure non trafiquée!
mov  ver_min_reel,bh   ; version mineure non trafiquée!
mov  revision,dl       ; révision.
mov  ver_flag,dh       ; flags de chargement.
non_supporte:          ; suite du programme.
```

Le switcher de tâches

Le DOSSHELL de la version 5.0 introduit un "switcher" ou commutateur de tâches qui permet d'avoir plusieurs applications chargées en même temps. Ce n'est pas un multitâche, ce qui signifie que seule l'application choisie tourne en mémoire, les autres étant gelées et swappées sur disque. Une application qui peut être sensible au changement de tâche (comme, par exemple, un programme de connexion sur un site central), a la possibilité de s'interfacer avec le switcher pour suivre ses opérations.

L'application doit tout d'abord tester si un switcher de tâches est présent (Int 2Fh/4B02h). Dans l'affirmative, il doit intercepter l'interruption 2Fh/4B01h qui est appelée par le switcher pour connaître les programmes désirant être tenus au courant de ses opérations. L'application communique alors en retour de cette interruption une adresse de notification, que le switcher appellera dorénavant pour signaler tout changement d'état tel que :

- l'initialisation et la mise en place du switcher, avec communication d'un point d'entrée pour les fonctions de service.
- la demande de commutation par l'utilisateur, qui peut être acceptée ou refusée par l'application.
- l'activation d'une nouvelle tâche.
- l'incorporation ou la destruction d'une entrée dans la liste des tâches.
- l'arrêt du switcher.

La version 5.0 est un événement important, d'autant qu'elle devrait se généraliser très rapidement sur l'ensemble des machines, faisant ainsi cesser la dualité DOS 3.3 / DOS 4.01. Outre les avantages qu'apporte cette nouvelle version aux utilisateurs, les programmeurs trouveront des fonctions leur facilitant l'écriture de programmes résidents ainsi que le chargement de ces derniers en mémoire haute.

Grâce à ces notifications envoyées par le switcher, l'application peut connaître exactement l'état du switcher. Elle peut également dialoguer avec lui au moyen de fonctions de service telles que :

- la demande de la version du switcher.
- le test d'une zone de mémoire pour savoir si elle est locale (donc switchée avec l'application) ou globale (donc non switchée et commune à toutes les applications).
- l'ajout ou le retrait d'un nouveau point d'entrée de notification. L'ensemble de ces fonctions spécifiques au switcher n'est pas simple, mais il a au moins le mérite d'exister et de permettre à une application sensible à la commutation de tâches, de coopérer avec le switcher.

Share

MS-DOS 5.0 intègre dorénavant certaines fonctions précédemment attribuées à l'utilitaire SHARE.EXE. Qui n'a pas déjà été intrigué par l'apparition du message «ATTENTION ! SHARE doit être chargé pour un support large» lors du boot d'un DOS 4 avec une partition supérieure à 32 Mo ? En fait, dès que le DOS 4 détecte une partition supérieure à 32 Mo, il teste la présence de l'utilitaire SHARE. Si celui-ci n'est pas chargé à partir du fichier CONFIG.SYS (par la commande INSTALL=SHARE.EXE) ou du fichier AUTOEXEC.BAT, le DOS 4 va même jusqu'à essayer de le charger automatiquement à partir du répertoire principal ou du répertoire indiqué par la variable d'environnement COMSPEC. En cas d'échec (si par exemple SHARE n'est pas dans l'un de ces répertoires), DOS 4 affiche alors le message d'avertissement précédemment indiqué.

```
Name          Size in Decimal     Size in Hex
------------  ------------------  ------------
SYSTEM          180256   (176.0K)     2C020
ANSI              4192   (  4.1K)      1060
SETVER             496   (  0.5K)       1F0
SMARTDRV         24896   ( 24.3K)      6140
KEYB              6208   (  6.1K)      1840
MOUSE            12784   ( 12.5K)      31F0
DOSKEY            3888   (  3.8K)       F30
FREE               192   (  0.2K)       C0
FREE             32496   ( 31.7K)      7EF0
FREE             62080   ( 60.6K)      F280

Total  FREE :    94768   ( 92.5K)

Total bytes available to programs (Conventional+Upper) :    710688   (694.0K)
Largest executable program size :                           615632   (601.2K)
Largest available upper memory block :                       62080   ( 60.6K)

   3145728 bytes total contiguous extended memory
         0 bytes available contiguous extended memory
   2059264 bytes available XMS memory
           MS-DOS resident in High Memory Area

C:\>_
```

Pourquoi tant d'insistance et quels sont les risques encourus en cas d'absence de SHARE ? Pour le comprendre, il faut revenir sur l'ancienne méthode d'accès des fichiers par FCB (File Control Block). Les valeurs contenues dans les différents champs d'un FCB ne permettent pas d'accéder à plus de 32 Mo.

Tout accès par FCB à un fichier contenu dans les premiers 32 Mo physiques du disque se passera sans problème. Mais, en cas d'accès à un fichier stocké après cette limite de 32 Mo, l'opération se fera alors à une adresse modulo 32 Mo.

Le danger d'une telle opération est flagrant : en cas de lecture, les données ne seront pas celles contenues dans le fichier, et en cas d'écriture, il y aura écrasement d'une portion du disque ! Bien sûr, ce dysfonctionnement ne se produit que pour des programmes utilisant l'ancienne méthode d'accès par FCB, remplacée depuis le DOS 2.0 par les accès par "handle". L'utilitaire SHARE implémentait en fait déjà un mécanisme de contrôle des FCB pour les accès multi-utilisateurs à un même fichier (fonctionnement en réseau ou en multitâche). Pour ce faire, il intercepte tous les appels FCB et travaille avec une copie locale et étendue du FCB, sur laquelle il peut effectuer tous les contrôles voulus. C'est donc tout naturellement que lui était échue la tâche de combler cette

Le fichier CONFIG.SYS s'étoffe en nouvelles commandes, axées sur le chargement en mémoire haute du DOS et des drivers.

lacune du DOS 4. Avec la version 5.0, Microsoft est revenu à une approche beaucoup plus pratique et plus sûre, en intégrant au noyau du système les opérations de contrôle des FCB sur les partitions 32 Mo. Le chargement de SHARE n'est donc plus obligatoire et ne doit se faire que dans les cas d'accès concurrents à un même fichier.

Disques et disquettes

MS-DOS 5.0 peut gérer les nouveaux lecteurs de disquette 2.88 Mo qui ne devraient pas tarder à équiper les machines haut de gamme. Très peu de modifications sont perceptibles au niveau du DOS lui-même, puisque la presque totalité du travail sera faite par le Bios de la machine. Le nouveau format est peu différent de celui du 1.44 Mo avec 2 faces, 80 pistes et 36 secteurs par piste au lieu de 18 pour le 1.44 Mo. Les fonctions Int 21h/440Dh/60h (Get Device Parameters) et Int 21h/440Dh/40h (Set Device Parameters) peuvent utiliser la valeur 08h dans le champ réservé au type du device afin de caractériser ce nouveau format. De même, la fonction Int 21h/440D/68h (Sense Media Type) peut rendre une valeur spécifique dans le cas d'un média 2.88 Mo.

MS-DOS 5.0 peut, désormais, également gérer jusqu'à huit unités de disques physiques, dans la mesure où l'Int 13h du Bios de la machine les gère également. Une telle configuration devrait être très rare, car les contrôleurs classiques ne sont prévus que pour deux unités maximum. La seule exception concerne le SCSI qui peut, lui, supporter jusqu'à sept disques. Une autre différence avec le DOS 4.0 est apparue avec la reconnaissance de plusieurs partitions primaires sur un même support. Malheureusement, le FDISK n'offre pas les possibilités de créer plusieurs partitions primaires et cela doit donc être réalisé à la main grâce à un éditeur de secteurs.

L'option /?

L'instauration de cette option en standard sur tous les utilitaires livrés avec le MS-DOS 5.0 n'apporte absolument aucun changement à l'interface de programmation du DOS. Mais elle ouvre la voie pour une standardisation de cette option, qu'il serait dommage de ne pas incorporer dans tout programme DOS en cours de développement.

En conclusion, les deux plus grandes modifications du DOS 5.0 par rapport à 4.0 sont essentiellement l'aspect gestion mémoire avec HMA et UMB, et la disponibilité de fonctions enfin "officielles" pour une gestion pratique des programmes résidents. Il faut aussi souligner l'excellent travail effectué sur le MS-DOS Programmer's Reference, d'un très

LA GESTION DES PROGRAMMES RÉSIDENTS (TSR)

Le principal problème rencontré jusqu'ici lors de l'écriture d'un TSR (programme résident appelé ainsi d'après les initiales de Terminate and Stay Resident), concernait la difficulté d'accès aux fonctions DOS du fait de son absence de réentrance. MS-DOS 5.0 comble cette lacune en documentant certaines fonctions "cachées" (non documentées).

La fonction Int 21h/34h permet de connaître, lors de l'installation du résident, l'adresse (segment:offset) de deux flags ou indicateurs spécifiant l'état du DOS. La règle est simple : si l'octet pointé par segment:offset+1 (ErrMode) est non nul, aucune fonction DOS ne peut être appelée. Dans le cas contraire, le test doit porter sur l'octet pointé par segment:offset (InDOS) qui, s'il est nul, permet l'appel de toutes les fonctions.

Si InDOS est non nul, le TSR peut cependant utiliser toutes les fonctions inférieures à 0Ch. Il peut maintenant arriver (en cas d'attente d'entrée au clavier par exemple), que l'indicateur InDOS soit en permanence positionné, interdisant certaines opérations DOS alors que la machine est en attente. Il est donc souhaitable que le résident intercepte l'Int 28h afin de déterminer les moments où le DOS est en attente, et où, malgré la présence du InDOS, il est possible d'appeler toutes les fonctions.

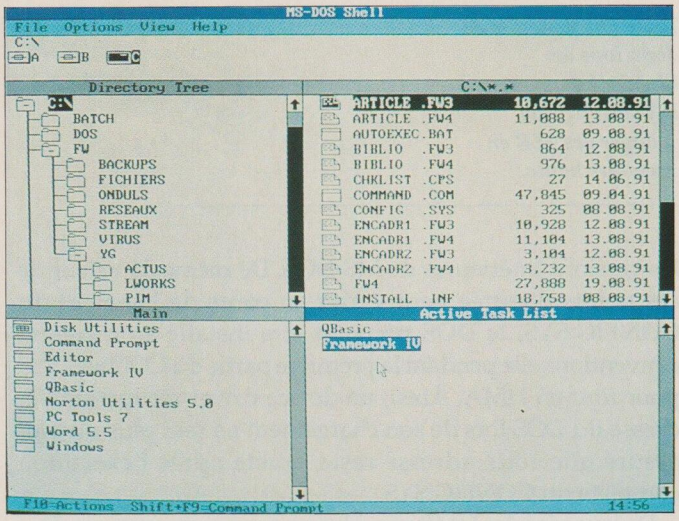
Il est également souhaitable d'intercepter l'Int 2Fh/1680h qui remplace désormais l'Int 28h et offre les mêmes possibilités. L'Int 2Fh est aussi une ressource appréciable pour un TSR. Avec un code fonction compris entre C0H ET FFh, elle permet de mettre en place des canaux de communication et d'interaction avec un TSR, sous une forme standardisée et compatible avec les résidents du DOS.

La fonction Int 21h/5D0Ah (Set Extended Error Information) permet de solutionner un problème particulier aux TSR. Lorsque celui-ci prend la main et exécute une fonction DOS qui génère une erreur étendue, les informations précédentes appartenant au programme interrompu sont alors perdues.

Un programme TSR correctement écrit, lorsqu'il appelle des fonctions DOS pouvant positionner ces informations d'erreur étendue, se doit au prélable de les sauvegarder (par l'Int 21h/59h), puis de les restaurer (par l'Int 21h/5DAh) avant de rendre la main.

Enfin, un TSR manipulant son PSP, doit au préalable sauvegarder (par l'Int 21h/51h) le PSP existant lors de son activation qui est celui du programme interrompu, donner l'adresse de son propre PSP (Int 21h/50h) puis restaurer de nouveau l'adresse du PSP initial (Int 21h/50h) avant de rendre la main. Comme on le voit, ces fonctions dorénavant documentées apportent un avantage certain pour l'écriture d'un programme TSR.

Il subsiste certes encore des zones d'ombre, et de nombreuses fonctions "cachées", mais un premier pas important a été effectué.



Le DOSSHELL réalise une commutation de tâches. Des fonctios de l'INT 21H permettent de tester si cette commutation est active ou pas.

haut niveau technique, d'une qualité de présentation irréprochable, et qui est désormais une référence incontournable pour le programmeur MS-DOS.

Le point sur la mémoire

La mémoire contenue dans une machine "compatible PC" peut se scinder en plusieurs parties :

- La mémoire conventionnelle. Elle est adressée de 00000h à 9FFFh, soit de 0 à 640 K. Accessible en mode réel, elle est disponible sur tous les types de processeurs employés dans les PC (du 8088 au i486). C'est la mémoire principalement utilisée par tous les logiciels (programmes d'application, device drivers, utilitaires TSR et bien sûr MS-DOS) et dont la taille figée se révèle de plus en plus pénalisante. L'un des principaux buts de MS-DOS 5.0 a été de minimiser son propre encombrement en mémoire conventionnelle et de fournir des outils et primitives de gestion mémoire pour les autres logiciels (device drivers, TSR et applications).
- La mémoire UMB (Upper Memory Blocks ou Blocs de mémoire supérieure). Elle s'adresse de 0A000h à 0FFFFh, c'est-à-dire entre 640 Ko et 1 Mo. C'est en fait un emplacement initialement réservé pour les extensions Bios (généralement en Prom) des cartes présentes dans la machine (vidéo, contrôleur disque, adaptateur réseau...). Cependant, il est rare que la totalité soit occupée et il devenait tentant de récupérer une partie de cet espace, qui est bien sûr directement accessible en mode réel, pour y charger des programmes et soulager la mémoire conventionnelle. Bien que les UMB puissent être envisagés sur tous les types de processeurs, seuls les 386 ou supérieurs disposent des fonctions de gestion mémoire suffisamment élaborées pour faire correspondre par logiciel de la mémoire vive RAM à ces adresses physiques réservées. C'est le rôle du driver EMM386.EXE qui peut être utilisé sur toute machine à base de 386 pour créer ces UMB. La disponibilité d'UMB sur 286 ou 8088/86 ne peut se faire que par des solutions matérielles, donc plus coûteuses et moins souples. La quantité de mémoire UMB dépend en fait des cartes présentes dans la machine et de l'espace occupé par leur Bios. Elle présente aussi la particularité de pouvoir être discontinue, offrant par exemple 32 K de C000h à C800h et 128 Ko de D000h à EFFFh.
- La mémoire étendue. Elle est adressée au-dessus de 10000h (1 Mo) sur toute machine à base de 286 ou supérieur. Cet espace s'étend jusqu'à la fin de la mémoire physiquement gérée par le processeur (16 Mo sur 286 et 4 Go sur 386/486). Aucune mémoire étendue n'est disponible sur 8088/86. La mémoire étendue n'étant accessible qu'en mode protégé, elle présente peu d'utilité sous DOS qui s'exécute exclusivement en mode réel. Elle n'est généralement exploitée que par des devices drivers (HIMEM, EMM386, RAMDRIVE, SMARTDRV...) ou par quelques programmes "étendus" tels que Lotus 1-2-3 3.0 ou Windows 3.0.
- La mémoire HMA (High Memory Area ou Zone de mémoire haute). Elle s'adresse de 10000h à 1FFEFh, soit dans les premiers 64 K (diminués de 16 octets) disponibles après la frontière du Méga. De même que la mémoire étendue, dont c'est le début, la HMA n'existe qu'à partir du 80286 et a son origine dans un "bug" de conception de ce chip. En théorie, le processeur ne peut adresser que 1 Mo en mode réel. Cependant, lorsqu'un segment est positionné à FFFFh et que l'offset varie de 0h à FFFFh, l'adresse physique réellement générée ne repasse pas à zéro mais dépasse le premier Méga-octet, permettant ainsi l'accès aux premiers 64 K de la mémoire étendue. Cette astuce a très vite été utilisée et permet d'accéder à 1 Mo + 64 Ko - 16 octets en mode réel.
- La mémoire EMS (Expanded Memory Specification), appelée aussi LIM (pour Lotus-Intel-Microsoft, les promoteurs de ce standard) ou mémoire paginée. D'une conception tout à fait différente, elle s'adresse en mode réel, à partir d'une fenêtre de 64 Ko, spécifiée entre 0C000h et 0EFFFh. Cette fenêtre permet d'adresser 8 Mo de mémoire dans sa version 3.2, et jusqu'à 32 Mo dans sa version 4.0. L'EMS peut être utilisée sur tous les types de processeurs, mais la lourdeur de la gestion par fenêtre ne permet d'y stocker que des données et non des pro-

```
C:\>type config.sys
DEVICE     = C:\DOS\SETVER.EXE
DEVICE     = C:\DOS\HIMEM.SYS
DEVICE     = C:\DOS\EMM386.EXE NOEMS
DOS        = HIGH,UMB
FILES      = 30
BUFFERS    = 10
COUNTRY    = 033,, C:\DOS\COUNTRY.SYS
DEVICEHIGH = C:\DOS\ANSI.SYS
DEVICEHIGH = C:\DOS\SMARTDRV.SYS 1024
STACKS     = 0,0
SHELL      = C:\DOS\COMMAND.COM /P /E:400

C:\>
C:\>ver/r

MS-DOS Version 5.00
Revision A
DOS is in HMA

C:\>_
```

Toute une panoplie de fonctions de l'Int 21H donnent la possibilité de charger des TSR en mémoire haute.

grammes. Sur un 386, le driver EMM386.EXE permet de créer cette mémoire EMS à partir de la mémoire étendue. Le MS-DOS 5.0 utilise très peu la mémoire EMS (RAMDRIVE, SMARTDRV et FASTOPEN uniquement).

Utilisation de la HMA et des UMB par DOS 5

La HMA n'est pas une notion très nouvelle. Elle était déjà gérée par le driver HIMEM.SYS suivant la spécification XMS 2.0 établie depuis plus de trois ans. Certains programmes l'utilisaient déjà pour s'y charger ou pour y stocker des données. L'une des principales nouveautés du MS-DOS 5.0 est d'avoir recours à la HMA pour charger la plus grande partie de son code, libérant ainsi une place précieuse en mémoire conventionnelle.

Pour cela, trois conditions sont nécessaires :

- la disponibilité de mémoire étendue (processeur 286 ou supérieur avec plus de 640 K de mémoire).
- la déclaration dans le fichier CONFIG.SYS d'un driver XMS aux normes 2.0 (HIMEM.SYS par exemple).
- la mention "dos=high" dans le fichier CONFIG.SYS.

En cas de succès, 45 Ko trouvent ainsi leur place au-dessus du Méga tandis que 28 K sont laissés en mémoire conventionnelle (essentiellement pour des raisons d'accès en fonction de la ligne A20 qu'il faut autoriser ou non). L'interface de programmation n'est pas affectée par le chargement du DOS en HMA. Seule l'information du lieu de chargement du DOS (HMA ou mémoire conventionnelle) devient disponible par l'Int 21H/3306h (Get MS-DOS Version). Cependant, la modification de la séquence de chargement du DOS 5 peut avoir des conséquences néfastes sur certains "device drivers" mal écrits. Lorsque la commande "dos=high" est incorporée au CONFIG.SYS, les points d'entrée du DOS peuvent être à une adresse supérieure à toutes les adresses de segment des device drivers, ce qui n'était jamais le cas avec les versions inférieures de MS-DOS. De même, la valeur de ces points d'entrée peut varier au cours de l'analyse du CONFIG.SYS, le DOS pouvant être installé en mémoire conventionnelle pendant la première partie du CONFIG.SYS pour finir en HMA. Ainsi, un device driver récupérant l'adresse du DOS lors de son chargement ne sera plus du tout assuré que cette adresse reste exacte après l'exécution complète du CONFIG.SYS.

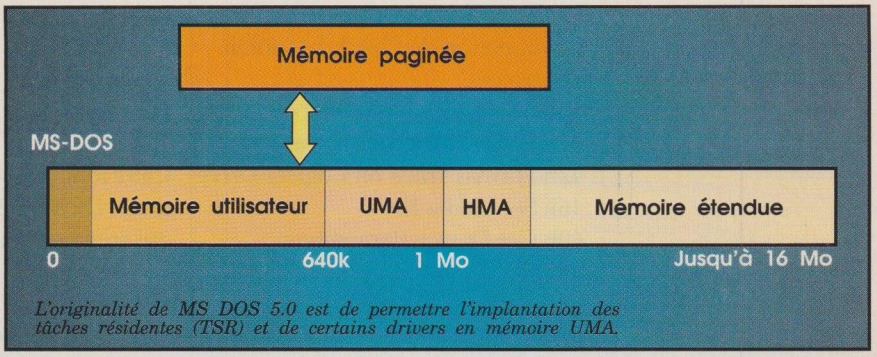
De leur côté, les UMB non plus ne sont pas nouveaux. Des utilitaires tels que QEMM ou 386MAX permettaient depuis longtemps de charger du code dans cette zone de mémoire. Le support par DOS 5 se fait au moyen de deux nouvelles commandes : "devicehigh" (utilisation dans le CONFIG.SYS pour le chargement des device drivers), et "loadhigh" ou "lh" pour les programmes exécutables. Bien entendu, la mention "dos=umb" doit être présente dans le fichier CONFIG.SYS pour que le DOS accepte son utilisation. L'interface de programmation s'enrichit de deux nouvelles fonctions (Int 21h/5802h et 21h/5803h), ainsi que de l'extension des deux fonctions gérant la stratégie d'allocation mémoire. Un programme souhaitant allouer des données en mémoire UMB doit donc suivre la procédure suivante :

- Sauvegarde de l'état initial (21h/5802h et 21h/5800h).
- Autorisation de l'allocation des UMB (21h/5803h).
- Modification de la stratégie d'allocation mémoire (21h/5801h).
- Allocation de la mémoire (21h/48h).
- Restauration de l'état initial (5803h et 5801h).

La sauvegarde/restauration de l'état initial est obligatoire afin de ne pas perturber les programmes exécutés à la suite, programmes qui pourraient ne pas être prévus pour s'installer dans un segment supérieur à A000h. Le chargement des programmes eux-mêmes en mémoire UMB est laissé à la discrétion de l'utilisateur qui peut se servir de la commande "loadhi" ou "lh". Pourraient être perturbés les programmes effectuant des calculs sur la valeur des segments, calculs qui pourraient mal se passer avec des segments supérieurs à A000h, mais aussi certains programmes cherchant à détecter leur installation résidente antérieure grâce à une recherche de signature en mémoire. Cette recherche se faisait généralement sur les adresses inférieures à celle de chargement, ce qui était cohérent avec la méthode d'allocation mémoire des versions de DOS antérieures.

Cela n'est plus viable avec le DOS 5.0 car un premier appel, fait avec loadhi, peut amener le programme à se charger en UMB, tandis que le chargement suivant sera effectué en mémoire conventionnelle (y compris si loadhi a été utilisé, dans

le cas où la mémoire est insuffisante). La solution dans ce cas consiste à utiliser l'interruption Multiplex 2Fh de la même manière que les TSR du MS-DOS, mais avec un code fonction compris entre C000h et FFFFh pour ne pas entrer en conflit avec eux. Toute nouvelle application doit désormais être capable soit de s'exécuter sans problème en mémoire UMB, soit de tester son adresse et d'afficher un message d'erreur en cas de chargement UMB.



L'originalité de MS DOS 5.0 est de permettre l'implantation des tâches résidentes (TSR) et de certains drivers en mémoire UMA.

Les devices drivers, quant à eux, doivent éviter de faire des suppositions sur leur adresse de chargement. Par exemple, certains produits requièrent que deux device drivers soient chargés à la suite, le deuxième estimant implicitement avoir été chargé à la suite du premier et pouvoir l'appeler directement. C'était certes toujours vrai pour les anciennes versions de DOS, mais cela ne l'est plus forcément avec la version 5.0. Le premier device driver peut être chargé en UMB, tandis que le deuxième, faute de place, pourra être placé en mémoire conventionnelle.

Enfin, un device driver pouvait toujours sans trop de problème "déborder" et manipuler de la mémoire après son adresse de fin physique, l'ensemble de la mémoire conventionnelle située à une adresse supérieure étant inoccupée. Ce n'est plus du tout le cas avec 5.0, puisqu'un device driver chargé en UMB peut avoir, à une adresse supérieure à la sienne, un autre device driver déjà chargé.

Il est donc très important qu'un device driver ne manipule plus la mémoire qui ne lui appartient pas, et pour cela MS-DOS 5.0 spécifie lors de l'appel d'initialisation du device driver (fonction 00h Init) l'adresse maximale qu'il a à sa disposition. Il devient donc de la responsabilité d'un device driver de tester la version du DOS, et si celle-ci est supérieure ou égale à 5.0, de tenir compte de cette valeur d'adresse maximale.

FONCTIONS NOUVELLES PAR RAPPORT À MS-DOS 4.0

Int 21h / 440Dh sous-fonction 68h (Generic IOCTL for Block Devices - Sense Media Type)

Permet de connaître le type de média pour un device de type bloc.

```
mov   ax,440Dh          ; numéro de fonction
mov   cx,0868h          ; numéro de sous-fonction
mov   bx,drive          ; numéro du drive
                        ; (0=défaut, 1=A, 2=B...)
mov   dx,offset buffer  ; ds : dx pointe sur un buffer
                        ; de 2 octets
int   21h               ; interruption DOS
jc    erreur            ; fonction non supportée ou media
                        ; indéterminé
```

Le premier octet du buffer reçu indique si le type de media correspond au media par défaut (1 dans ce cas, 0 dans le cas contraire). Le second octet indique le type de media (02h pour 720K, 07h pour 1.44M et 09h pour 2.88M). L'implémentation de cette sous-fonction peut être testée grâce à la fonction Int 21h/4411h *(voir plus bas)*.

Int 21h / 4410h (Query IOCTL Handle)

Permet de vérifier si certaines des sous-fonctions de l'Int 21h/440Ch sont supportées par un device driver (de type caractère).

```
mov ax,4410h          ; numéro de fonction
mov bx,handle         ; handle du device à contrôler
mov ch,categorie      ; catégorie du device à contrôler :
                      ;  01h pour un device série
                      ;  03h pour la console écran
                      ;  05h pour l'imprimante parallèle
mov cl,fonction       ; sous-fonction de l'Int 21h/440Ch
                        à contrôler :
                      ;  45h Set Iteration Count
                      ;  65h Get Iteration Count
int  21h              ; interruption DOS
jnc  support          ; pas de carry si la fonction
                        est supportée
```

Il est souhaitable d'utiliser dorénavant cette fonction avant d'exécuter une sous-fonction 45h ou 65h sur un device caractère.

Int 21h / 4411h (Query IOCTL Device)

Permet de vérifier si les fonctions de l'Int 21h/440Dh sont supportées par un device driver (de type bloc).

```
mov   ax,4411h          ; numéro de fonction
mov   bl,drive          ; drive (0=défaut, 1=1, 2=B...)
mov   ch,8              ; catégorie du device à contrôler
```

```
                        (8 obligatoire)
mov   cl,fonction       ; sous-fonction de l'Int 21h/440Dh
                          à contrôler :
                        ;  40h Set Device Parameters
                        ;  41h Write Track on Logical Drive
                        ;  42h Format Track on Logical Drive
                        ;  46h Set Media ID
                        ;  60h Get Device Parameters
                        ;  61h Read Track on Logical Drive
                        ;  62h Verify Track on Logical Drive
                        ;  66h Get Media ID
                        ;  68h Sense Media Type
int   21h               ; interruption DOS
jnc   support           ; pas de carry si la fonction est
                          supportée
```

Il est souhaitable d'utiliser dorénavant cette fonction avant d'exécuter tout appel à l'Int 21h/440Dh.

Int 21h / 5802h (Get Upper-Memory Link)

Permet de connaître l'état de l'autorisation d'allocation de mémoire UMB :

```
mov   ax,5802h          ; numéro de fonction
int   21h               ; interruption DOS
jc    erreur            ; fonction inexistante avant
                          la version 5.0 du DOS
mov   resultat,al       ; 1 si l'allocation UMB est autorisée
                        ; 0 si l'allocation UMB n'est pas
                          autorisée
```

Int 21h / 5803h (Set Upper-Memory Link)

Permet de modifier l'autorisation d'allocation de mémoire des UMB :

```
mov   ax,5803h          ; numéro de fonction
mov   bx,autorisation   ; 1 pour autorisation
                        ; 0 pour interdiction
jc    erreur            ; fonction inexistante avant
                          la version 5.0 du DOS
```

Il est recommandé de toujours sauvegarder l'état initial de l'autorisation d'allocation avant de la modifier, puis de la restaurer avant de rendre la main au système.

Int 21h / 4B05h (Set Execution State)

Permet de préparer un programme pour son exécution. Cette préparation consiste essentiellement à déterminer le numéro de version DOS qui doit lui être renvoyé (dans le cas où SETVER a été activé).

```
mov   ax,4B05h          ; numéro de fonction
mov   dx,offset execstate ; ds:dx pointe sur une structure
                            de type EXECSTATE
int   21h               ; interruption DOS
jc    erreur            ; en cas d'erreur
```

Cette fonction semble être essentiellement utilisée en interne par le DOS. Elle doit être aussi utilisée en cas de programme qui intercepte la fonction Int 21h/4B00 (Load and Execute Program).

Int 21h / 3306h (Get MS-DOS Version)

Permet de connaître la version réelle du MS-DOS, ainsi que son lieu de chargement.

```
mov   ax,3306h          ; numéro de fonction
int   21h               ; interruption DOS
cmp   al,0FFh           ; fonction non supportée sur
                          un DOS de version
je    non_supporte      ; inférieure à 5.0
mov   ver_majeure,bl    ; version majeure
                          (5 pour le MS-DOS 5.0)
mov   ver_mineure,bh    ; version mineure
                          (0 pour le MS-DOS 5.0)
mov   revision,dl       ; révision dans les 3 LSB
                          (bits 0 à 2) de dl
                        ;  000b pour la version actuelle
                          (révision A)
mov   ver_flag,dh       ; lieu de chargement :
                        ;  00h : mémoire conventionnelle
                        ;  10h : mémoire HMA
                        ;  08h : MS-DOS est en ROM
```

Cette fonction devient primordiale, dans la mesure où le résultat de la fonction Int 21h/30h (Get Version Number) peut être complètement manipulé par l'utilisateur grâce à l'utilitaire SETVER !!

FONCTIONS MODIFIÉES PAR RAPPORT À MS-DOS 4.0

Int 21h / 5800h (Get Allocation Strategy)

Permet de connaître la stratégie d'allocation mémoire utilisée par le DOS :

```
mov   ax,5800h          ; numéro de fonction
int   21h               ; interruption DOS
mov   strategie,ax      ; stratégie utilisée.
```

Parmi les différentes valeurs possibles pour désigner la stratégie, les trois premières existaient déjà. Seules les six der-

nières sont spécifiques à la version 5 du DOS, en raison de l'utilisation des UMB.

0000h (FIRST_FIT_LOW) pour allocation en mémoire conventionnelle à partir de l'adresse la plus basse.
0001h (BEST_FIT_LOW) pour allocation en mémoire conventionnelle dans le bloc le plus adapté.
0002h (LAST_FIT_LOW) pour allocation en mémoire conventionnelle à partir de l'adresse la plus haute.
0040h (FIRST_FIT_HIGHONLY) pour allocation en UMB à partir de l'adresse la plus basse.
0041h (BEST_FIT_HIGHONLY) pour allocation en UMB dans le bloc le plus adapté.
0042h (LAST_FIT_ONLY) pour allocation en UMB à partir de l'adresse la plus haute.
0080h (FIRST_FIT_HIGH) pour allocation en mémoire UMB à partir de l'adresse la plus basse. En cas d'insuccès, la mémoire conventionnelle est alors utilisée.
0081h (BEST_FIT_HIGH) pour allocation en mémoire UMB dans le bloc le plus adapté. En cas d'insuccès, la mémoire conventionnelle est alors utilisée.
0082h (LAST_FIT_HIGH) pour allocation en mémoire UMB à partir de l'adresse la plus haute. En cas d'insuccès, la mémoire conventionnelle est alors utilisée.

Int 21h / 5801h (Set Allocation Strategy)

Permet de modifier la stratégie d'allocation mémoire utilisée par le DOS :

```
mov   ax,5801h        ; numéro de fonction
mov   bx,strategie    ; nouvelle stratégie d'allocation
                        mémoire
int   21h             ; interruption DOS
jc    erreur          ; erreur en cas de stratégie inconnue
```

Les valeurs représentant la stratégie sont les mêmes que pour la fonction précédente. Il est recommandé de toujours sauvegarder l'état initial de la stratégie d'allocation avant de la modifier, puis de la restaurer avant de rendre la main au système.

FONCTIONS NOUVELLEMENT DOCUMENTÉES

Ce sont des fonctions déjà existantes dans la version 4.0 ou même dans une version antérieure, mais qui n'ont jamais été documentées par Microsoft. Elles constituent une partie des fonctions "cachées" du MS-DOS. Elles sont désormais officialisées, ce qui signifie qu'elles seront désormais supportées dans les futures versions.

Int 21h / 440Dh sous-fonction 66h (Generic IOCTL for Block Devices - Get Media ID)

Permet de connaître le label du volume, le numéro de série et le système de fichier utilisé sur un drive.

```
mov   ax,440Dh          ; numéro de fonction.
mov   cx,0866h          ; numéro de sous-fonction.
mov   bx,drive          ; drive (0=défaut, 1=A, 2=B, etc).
mov   dx,offset MediaID ; DS:DX pointe sur un buffer
                          ; de 25 octets
int   21h               ; interruption DOS
jc    erreur            ; en cas d'erreur
```

Le buffer contient les informations sous la forme :

* 00h : réservé à zéro (deux octets).
* 02h : numéro de série (quatre octets).
* 06h : label du volume (onze caractères).
* 11h : système de fichier (huit caractères) qui peut être FAT12 ou FAT16.

Int 21h / 440Dh sous-fonction 46h (Generic IOCTL for Block Devices - Set Media ID)

Permet de modifier le label du volume, le numéro de série et le système de fichier utilisé sur un drive.

```
mov   ax,440Dh          ; numéro de fonction.
mov   cx,0846h          ; numéro de sous-fonction.
mov   bx,drive          ; drive (0=défaut, 1=A, 2=B, etc).
mov   dx,offset MediaID ; DS:DX pointe sur un buffer qui
                          contient les informations
                          nécessaires sous la même forme
                          que la fonction Int 21h/440Dh/66h.
int   21h               ; interruption DOS
jc    erreur            ; en cas d'erreur
```

Si le label du volume et le numéro de série peuvent être changés sans problème, il semble cependant beaucoup plus aléatoire et risqué de modifier le type du système de fichier utilisé ! Il sera donc souhaitable, avant toute modification, de sauvegarder sa valeur afin de pouvoir la conserver lors d'une modification des autres paramètres.

Int 21h / 1Fh (Get Default DPB)

Permet de connaître l'adresse du DPB (Disk Parameter Block) du drive courant.

```
mov   ah,1Fh            ; numéro de fonction
int   21h               ; interruption DOS
cmp   al,0FFh
jz    erreur            ; erreur
```

DS:BX pointe alors sur le DPB du disque courant qui contient les informations relatives au drive (et bien sûr au media qui y est inséré dans le cas des lecteurs de disquette) :

00h - 00h : numéro du drive (0=A, 1=B, etc).
01h - 01h : numéro d'unité du drive (un même device driver peut gérer plusieurs unités).
02h - 03h : taille des secteurs en octets.
04h - 04h : nombre de secteurs par unité d'allocation diminué de 1.
05h - 05h : nombre de secteurs par unité d'allocation, exprimé en puissance de 2 (2 pour 4, 3 pour 8, etc).
06h - 07h : numéro du premier secteur de la FAT (table d'allocation).
08h - 08h : nombre de FAT (table d'allocation) sur ce drive.
09h - 0Ah : nombre d'entrées possibles dans le répertoire principal.
0Bh - 0Ch: numéro du premier secteur de la première unité d'allocation.
0Dh - 0Eh: nombre d'unités d'allocation + 1 sur ce drive.
0Fh - 10h : nombre de secteurs occupés par une FAT (table d'allocation).
11h - 12h : numéro du premier secteur contenant le répertoire principal.
13h - 16h : adresse physique du device driver (segment:offset).
17h - 17h : descripteur de media.
18h - 18h : indique si le drive a déjà été accédé (0FFh sinon).
19h - 1Ch : adresse du DPB suivant (segment:offset). Les DPB sont chaînés.
1Dh - 1Eh : dernière unité d'allocation allouée.
1Fh - 20h : nombre d'unités d'allocation libres.

Cette fonction existait depuis la version 2.0, mais n'avait jamais été documentée. Elle est largement employée par un grand nombre d'utilitaires car elle permet de connaître des informations sur le drive courant, qui ne peuvent pas être obtenues autrement. Son utilisation sous des versions de MS-DOS inférieures à 5.0 doit être faite avec précaution car la structure des informations n'est pas restée constante tout au long de l'évolution des versions. Si la structure utilisée par le DOS 4.x semble identique à celle du 5.0, il n'en va pas de même pour le DOS 3.x par exemple, qui ne consacre qu'un seul octet au nombre de secteurs occupés par une FAT au lieu de 2 pour les versions 4 et 5. Il en résulte que toutes les informations suivantes (à partir de l'offset 11h) sont décalées d'un octet ! En tout état de cause, Microsoft ne garantit son résultat qu'à partir du DOS 5.0.

Int 21h / 32h (Get DPB)

Permet de connaître l'adresse du DPB (Disk Parameter Block) pour un drive quelconque.

```
mov   ah,32h          ; numéro de fonction
mov   dl,drive        ; numéro du drive
                      ; (0=défaut, 1=A, etc)
int   21h             ; interruption DOS
cmp   al,0FFh
jz    erreur          ; erreur
```

DS:BX pointe alors sur le DPB du disque spécifié. Les informations sont identiques à celles de la fonction Int 21h/1Fh.

Int 21h / 6Ch (Extended Open/Create)

Permet de combiner en un seul appel, les fonctions Int 21h/3Ch (Create File with Handle), Int 21h/3Dh (Open File with Handle) et Int 21h/68h (Commit File).

```
mov   ah,6Ch          ; numéro de fonction
mov   bx,mode_acces   ; modes d'accès au fichier.
                      ; Ces modes comprennent ceux
                      ; de l'Int 21h/3Dh et deux nouveaux
                      ; modes (voir ci-dessous).
mov   cx,attributs    ; attributs du fichier s'il doit être
                      ; créé. Identiques à ceux de
                      ; l'Int 21h/3Ch.
mov   dx,action       ; action à entreprendre suivant la
                      ; présence ou l'absence du fichier.
mov   si,offset nom   ; DS:SI doit pointer sur le nom du
                      ; fichier.
int   21h             ; interruption DOS
jc    erreur          ; en cas d'erreur
```

La valeur du registre DX est l'action à entreprendre, et peut être une combinaison des valeurs suivantes :

* 0001h : FILE_CREATE : Crée un nouveau fichier. Rend une erreur si celui-ci existe déjà.

* 0010h : FILE_OPEN : Ouvre seulement le fichier. Rend une erreur si celui-ci n'existe pas.

* 0020h : FILE_TRONCATE : Remplace un fichier existant.

Deux nouveaux modes sont autorisés et peuvent être combinés avec ceux déjà existants dans la fonction Int 21h/3Dh :

* 2000h : OPEN_FLAGS_NOCRIT_ERR : permet de ne pas appeler le handler d'erreur critique (Int 24h) lorsqu'une erreur critique se produit sur ce fichier, mais de retourner cette erreur directement au programme.
C'est une possibilité très intéressante dans le cas d'accès de fichiers sur un lecteur de disquette. Un message «Veuillez insérer la disquette...» sera quand même plus agréable que le barbare «Unité ou périph. non prêt. Abandon, Reprise, Echec ?».

* 4000h : OPEN_FLAGS_COMMIT : force la mise à jour obligatoire du fichier après chaque écriture. Les buffers internes du MS-DOS sont réécrits.

Cette fonction existait depuis la version 4.0 mais n'était pas documentée jusqu'ici. Elle devrait permettre une gestion plus souple et plus simple de l'ouverture/création des fichiers.

Int 21h / 34h (Get InDOS Flag Address)

Permet de connaître l'adresse de l'indicateur "InDOS", qui indique si le DOS est actuellement en cours d'exécution.

```
mov   ah,34h              ; numéro de fonction
int   21h                 ; interruption DOS
mov   al,byte ptr es:[bx] ; ES:BX indique l'adresse de
                          ; l'indicateur
```

Cette fonction est la plus connue et la plus utilisée des fonctions dites "cachées" du MS-DOS. Elle existe depuis la version 2.0 du MS-DOS et permet aux programmes résidents d'utiliser les fonctions du DOS malgré sa non-réentrance. L'indicateur "InDOS" est incrémenté quand le DOS commence le traitement d'une interruption 21h. Il est décrémenté lorsqu'il en sort. Il est possible de savoir de la sorte si le MS-DOS est en cours d'exécution (auquel cas il est dangereux d'utiliser certains de ses services), ou s'il n'est pas utilisé (indicateur à 0). Dans le cas où InDOS est positionné, seules les fonctions de 01h à 0Ch peuvent être appelées. Il est toutefois souhaitable de traiter simultanément l'indicateur "ErrMode" qui spécifie si le MS-DOS traite une erreur critique, auquel cas aucune fonction ne doit être appelée (même celles inférieures à 0Ch). L'indicateur "ErrMode" est l'octet situé juste avant "InDOS" (à partir de la version 3.x). A noter que la fonction 21h/34h ne doit être appelée qu'une seule fois au départ, et que tout accès ultérieur aux indicateurs doit se faire directement à l'adresse reçue.

Int 21h / 4B01h (Load Program)

Permet de charger un programme en mémoire sans l'exécuter.

```
mov   ax,4B01h            ; numéro de fonction.
mov   dx,offset nom       ; DS:DX pointe sur le nom du
                          ; programme.
mov   bx,offset load      ; ES:BX pointe sur une structure
                          ; d'informations de chargement.
int   21h                 ; interruption DOS.
jc    erreur              ; en cas d'erreur.
```

La structure pointée par ES:BX reprend les mêmes informations que celles nécessaires pour la fonction Int 21h/4B00h (Load and Execute Program) mais comprend à sa suite 8 octets qui ne servent pas lors de l'appel, mais qui sont mis à jour à la suite du chargement :

* 0Eh : adresse de début du code (segment:offset).
* 12h : adresse du début de la pile (segment:offset).

Cette fonction existe depuis la version 2.0. Elle est souvent utilisée par les "debuggers" qui peuvent ainsi charger un programme avant de l'exécuter pas à pas.

Int 21h / 51h (Get PSP Address)

Permet de connaître l'adresse du PSP (Program Segment Prefix) du code exécuté.

```
mov   ax,51h              ; numéro de fonction
int   21h                 ; interruption DOS
mov   segmentpsp,bx       ; BX contient le segment du PSP
```

Le PSP est la zone de 256 octets qui est allouée avant chaque programme et qui contient, parmi d'autres informations, l'adresse de son environnement. Sans cette adresse, il était difficile jusqu'ici à un programme de connaître son environnement. Cette fonction existe depuis la version 2.0.

Int 21h / 62h (Get PSP Address)

Cette fonction est absolument identique à la précédente.

Int 21h / 50h (Set PSP Address)

Permet de modifier l'adresse du PSP (Program Segment Prefix) du code exécuté.

```
mov   ax,50h              ; numéro de fonction
mov   bx,seg_psp          ; BX contient le nouveau segment
                          ; du PSP
int   21h                 ; interruption DOS
```

Cette fonction existe depuis la version 2.0.

Int 21h / 5D0Ah (Set Extended Error)

Permet de spécifier les paramètres qui seront renvoyés lors du prochain appel de la fonction Int 21h/59h (Get Extended Error).

```
mov   ax,5D0Ah            ; numéro de fonction
mov   si,offset Err_info  ; DS:SI pointe sur la zone de 22 octets
                          ; contenant la valeur des registres
                          ; qui seront renvoyés.
int   21h                 ; interruption DOS
```

Cette fonction existe depuis la version 3.1 mais n'est supportée par Microsoft qu'à partir de 4.0.

Int 21h / 6505h (Get Filename-Character Table)

Permet de connaître les caractères qui ne doivent pas être utilisés dans les noms de fichiers, en fonction du "Country" et du "Code page".

```
mov   ax,6505h          ; numéro de fonction.
mov   bx,codepage       ; codepage souhaité (850 pour la
                        ; France). Si 0FFFFh, le codepage
                        ; courant est utilisé.
mov   dx,country        ; pays souhaité (033 pour la France).
                        ; Si 0FFFFh, le country courant est
                        ; utilisé.
mov   cx,5              ; taille du buffer (5 octets minimum).
mov   di,offset buffer  ; ES:DI pointe sur le début du buffer.
int   21h               ; interruption DOS.
jc    erreur            ; erreur si CARRY positionné.
```

Au retour de la fonction, le buffer contient un octet égal à 05h (identifiant) et l'adresse complète (segment:offset) de la table des caractères interdits. Cette fonction est supportée à partir de la version 3.3.

Int 21H / 6520h (Convert Character)

Permet de convertir un caractère en majuscule.

```
mov   ax,6520h          ; numéro de fonction.
mov   dl,caractere      ; caractère à convertir.
int   21h               ; interruption DOS.
jc    error             ; en cas d'erreur.
```

Après l'exécution de la fonction, le registre DL contient le caractère en majuscule. Cette fonction est supportée à partir de la version 3.3.

Int 21h / 6521h (Convert String)

Permet de convertir en majuscules une chaîne de caractères de longueur donnée.

```
mov   ax,6521h          ; numéro de fonction.
mov   dx,offset chaine  ; DS:DX pointe sur le 1er caractère.
mov   cx,longueur       ; longueur de la chaîne.
int   21h               ; interruption DOS.
jc    error             ; en cas d'erreur.
```

Après l'exécution de la fonction, la chaîne de caractères a été transcodée en majuscules. Cette fonction est supportée à partir de la version 3.3.

Int 21h / 6522h (Convert ASCIIZ String)

Permet de convertir en majuscules une chaîne de caractères terminée par 00.

```
mov   ax,6522h          ; numéro de fonction.
mov   dx,offset chaine  ; DS:DX pointe sur le premier
                        ; caractère d'une chaîne terminée
                        ; par 00.
int   21h               ; interruption DOS.
jc    error             ; en cas d'erreur.
```

Après l'exécution de la fonction, le registre DL contient le caractère en majuscule (si bien sûr la transformation était applicable au caractère choisi). Cette fonction est supportée à partir de la version 3.3.

FONCTIONS REMPLACÉES

L'évolution permanente du MS-DOS au cours des dix dernières années a conduit à doubler certaines fonctions par des équivalents disposant d'une meilleure "compatibilité" avec les versions futures ainsi que d'une meilleure sécurité. Il s'agit bien sûr de fonctions issues de la version 1.0 et dont les équivalents étaient présents dès la version 2.0 en 1983 ! Le DOS 4.0 listait déjà 17 fonctions. Le DOS 5.0 en rajoute 9 autres qu'il faudra donc éviter d'utiliser à l'avenir. Le programmeur averti ne sera pas surpris d'y retrouver toutes les entrées/sorties directes, auxquelles sont préférées dorénavant les opérations à base de handle prédéfinis.

Int 21h / 01h (Read Keyboard with Echo)

Permet de lire un caractère à partir du clavier. Le caractère est affiché à l'écran et le Control-C est actif.

```
mov   ah,01h            ; numéro de fonction.
int   21h               ; interruption du DOS.
mov   caractere,al      ; caractère reçu dans le registre AL
```

Cette fonction doit être systématiquement remplacée par la fonction Int 21h/3Fh (Read File or Device) utilisée avec le handle 0 (STDIN).

```
mov   ah,3Fh            ; numéro de fonction.
mov   dx,offset buffer  ; DS:DX pointe sur le buffer de
                        ; réception
mov   cx,1              ; 1 seul caractère à entrer.
mov   bx,0              ; handle 0 = entrée standard.
int   21h               ; interruption DOS
jc    error             ; en cas d'erreur.
```

S'il est important que l'entrée se fasse à partir du clavier et ne puisse pas être redirigée, il suffit d'ouvrir par la fonction Int 21h/3Ch le device CON et d'utiliser le handle obtenu à la place du handle 0.

Int 21h / 02h (Display Character)

Permet d'afficher un caractère à l'écran.

```
mov   ah,02h          ; numéro de fonction.
mov   dl,caractere    ; caractère à afficher.
int   21h             ; interruption DOS.
```

Cette fonction doit être systématiquement remplacée par la fonction Int 21h/40h (Write File or Device) utilisée avec le handle 1 (STDOUT).

```
mov   ah,40h          ; numéro de fonction.
mov   dl,caractere    ; caractère à afficher.
mov   byte ptr buffer,dl
mov   dx,offset buffer ; DS : DX pointe sur le buffer.
mov   cx,1            ; 1 seul caractère à entrer.
mov   bx,1            ; handle 1 = sortie standard.
int   21h             ; interruption DOS
jc    error           ; en cas d'erreur.
```

S'il est important que la sortie se fasse à l'écran et ne puisse pas être redirigée, il suffit d'ouvrir par la fonction Int 21h/3Ch le device CON et d'utiliser le handle obtenu à la place du handle 1.

Int 21h / 03h (Auxiliary Input)

Permet de lire un caractère sur l'entrée auxiliaire.

```
mov   ah,03h          ; numéro de fonction.
int   21h             ; interruption du DOS.
mov   caractere,al    ; caractère reçu dans le registre AL
```

Cette fonction doit être systématiquement remplacée par la fonction Int 21h/3Fh (Read File or Device) utilisée avec le handle 3 (STDAUX).

```
mov   ah,3Fh          ; numéro de fonction.
mov   dx,offset buffer ; DS:DX pointe sur le buffer de
                      ;  réception
mov   cx,1            ; 1 seul caractère à entrer.
mov   bx,3            ; handle 3 = auxiliaire.
int   21h             ; interruption DOS.
jc    error           ; en cas d'erreur.
```

Int 21h / 04h (Auxiliary Output)

Permet de sortir un caractère sur la sortie auxiliaire.

```
mov   ah,04h          ; numéro de fonction.
mov   dl,caractere    ; caractère à sortir.
int   21h             ; interruption du DOS.
```

Cette fonction doit être systématiquement remplacée par la fonction Int 21h/40h (Write File or Device) utilisée avec le handle 3 (STDAUX).

```
mov   ah,40h          ; numéro de fonction.
mov   dl,caractere    ; caractère à afficher.
mov   byte ptr buffer,dl
mov   dx,offset buffer ; DS:DX pointe sur le buffer.
mov   cx,1            ; 1 seul caractère à sortir.
mov   bx,3            ; handle 3 = auxiliaire.
int   21h             ; interruption DOS.
jc    error           ; en cas d'erreur.
```

Int 21h / 05h (Print Character)

Permet de sortir un caractère sur l'imprimante.

```
mov   ah,05h          ; numéro de fonction.
mov   dl,caractere    ; caractère à sortir.
int   21h             ; interruption du DOS.
```

Cette fonction doit être systématiquement remplacée par la fonction Int 21h/40h (Write File or Device) utilisée avec le handle 4 (STDPRN).

```
mov   ah,40h          ; numéro de fonction.
mov   dl,caractere    ; caractère à imprimer.
mov   byte ptr buffer,dl
mov   dx,offset buffer ; DS:DX pointe sur le buffer.
mov   cx,1            ; 1 seul caractère à imprimer.
mov   bx,4            ; handle 4 = imprimante standard.
int   21h             ; interruption DOS.
jc    error           ; en cas d'erreur.
```

Int 21h / 09h (Display String)

Permet d'afficher une chaîne de caractères terminée par un signe dollar ('$').

```
chaîne db "test","$"
...

mov   ax,09h          ; numéro de fonction.
mov   dx,offset chaine ; DS:DX pointe sur la chaîne
int   21h             ; interruption DOS
```

Cette fonction doit être systématiquement remplacée par la fonction Int 21h/40h (Write File or Device) utilisée avec le handle 1 (STDOUT).

```
chaîne    db  "test"
longueur  equ $-chaîne
...
mov   ah,40h          ; numéro de fonction.
mov   dx,offset chaine ; DS:DX pointe sur la chaîne de
                      ;  caractère.
mov   cx,longueur     ; nombre de caractères à afficher.
```

```
mov   bx,1          ; handle 1 = sortie standard.
int   21h           ; interruption DOS.
jc    error         ; en cas d'erreur.
```

Int 21h / 0Ah (Buffered Keyboard Input)

Permet de lire plusieurs caractères à partir du clavier. Les caractères sont affichés à l'écran et le Control-C est actif.

```
mov   ah,0Ah            ; numéro de fonction.
mov   dx,offset buffer  ; DS:DX pointe sur le buffer de
                        ;  réception.
mov   al,longueur       ; longueur maximum à entrer.
mov   byte ptr buffer[0],al
int   21h               ; interruption du DOS.
                        ; Au retour, le deuxième octet du
                        ; buffer indique le nombre de
                        ; caractères entrés, non compris
                        ; le retour chariot de la fin.
```

Cette fonction doit être systématiquement remplacée par la fonction Int 21h/3Fh (Read File or Device).

```
mov   ah,3Fh            ; numéro de fonction.
mov   dx,offset buffer  ; DS:DX pointe sur le buffer de
                        ; réception
mov   cx,longueur       ; longueur maximum à entrer.
mov   bx,0              ; handle 0 = entrée standard.
int   21h               ; interruption DOS
jc    error             ; en cas d'erreur.
                        ; au retour, ax contient le nombre
                        ; de caractères entrés, y compris
                        ; le retour chariot et le saut de ligne.
```

S'il est important que l'entrée se fasse à partir du clavier et ne puisse pas être redirigée, il suffit d'ouvrir par la fonction Int 21h/3Ch le device CON et d'utiliser le handle obtenu à la place du handle 0.

Int 21h / 1Bh (Get Default Drive Data)

Permet de connaître le nombre de secteurs par unité d'allocation, le nombre d'octets par secteur, le nombre d'unités d'allocation et le descripteur de media, pour le drive courant.

```
mov   ah,1Bh        ; numéro de fonction.
int   21h           ; interruption DOS.
cmp   al,0FFh       ; en cas d'erreur.
je    erreur        ; AL = secteurs par unité d'allocation.
                    ; CX = octets par secteur.
                    ; DX = nombre d'unités d'allocation.
                    ; DS:BX pointe sur le descripteur
                    ; de media.
```

Cette fonction doit être systématiquement remplacée par la fonction Int 21h/36h (Get Disk Free Space).

```
mov   ah,36h        ; numéro de fonction.
mov   dl,drive      ; drive (0=défaut, 1=A, 2=B, etc).
int   21h           ; interruption DOS.
cmp   ax,0FFFFh     ; en cas d'erreur.
je    erreur        ; AX = secteurs par unité d'allocation.
                    ; CX = octets par secteur.
                    ; DX = nombre total d'unités
                    ; d'allocation.
                    ; BX = nombre d'unités d'allocations
                    ; libres.
```

Si l'on souhaite récupérer également le descripteur de media, la fonction Int 21h/32h (Get DPB) pourra être utilisée à partir de la version 5.0.

Int 21h / 1Ch (Get Drive Data)

Permet de connaître le nombre de secteurs par unité d'allocation, le nombre d'octets par secteur, le nombre d'unités d'allocation et le descripteur de media, pour un drive courant. C'est l'extension de la fonction précédente, le drive étant donné dans le registre DL (0=défaut, 1=A, 2=B, etc). Cette fonction doit être systématiquement remplacée par la fonction Int 21h/36h (Get Disk Free Space).

INTERRUPTIONS REMPLACÉES

De même que certaines fonctions de l'Int 21h ont été remplacées par des équivalents, certaines interruptions ont été également "doublées". C'est le cas par exemple de l'interruption 20h (Terminate Program) remplacée par l'Int 21h/4Ch, et de l'interruption 27h (Terminate and Stay Resident) remplacée par l'Int 1h/31h. MS-DOS 5.0 en ajoute deux autres à la liste, mais non des moindres, puisqu'il s'agit des interruptions 25h et 26h, utilisées pour la lecteur directe des secteurs d'un drive.

INT 25h (Absolute Disk Read)

Permet de lire un ou plusieurs secteurs logiques à partir du drive spécifié.

```
mov   al,drive          ; drive (0=A, 1=B, etc).
mov   bx,offset buffer  ; DS:BX pointe sur le buffer de
                        ;  réception.
mov   cx,nb_secteurs    ; nombre de secteurs à lire.
mov   dx,no_secteur     ; numéro du premier secteur.
int   25h               ; interruption DOS.
jc    erreur            ; en cas de problème.
popf                    ; attention aux flags de retour !
```

L'interface de cette interruption avait déjà largement été modifiée par l'apparition du DOS 4.0 avec ses partitions supérieures à 32 Mo, puisque le numéro de secteur pouvait dépasser 65536 et ne tenait donc plus dans le registre DX. Le nouvel accès devait se faire à partir d'une structure complémentaire adressée par DS:BX.

```
mov   al,drive          ; drive (0=A, 1=B, etc).
mov   bx,offset info    ; DS:BX pointe sur les informations
                        ; complémentaires.
mov   cx,0FFFFh         ; indique la présence des informations
                        ; complémentaires.
int   25h               ; interruption DOS.
jc    erreur            ; en cas de pròblème.
popf                    ; attention aux flags de retour !
```

La structure complémentaire comprend 10 octets répartis comme suit :

* 00h : numéro du premier secteur à lire sur 4 octets.
* 04h : nombre de secteurs à lire sur 2 octets.
* 06h : adresse complète du buffer de lecture (segment:offset).

La documentation du DOS 5.0 recommande l'utilisation de la fonction de l'Int 21h/440Dh/61h (Read Track on Logical Drive), existante depuis le DOS 3.2, en lieu et place de l'interruption 25h. En fait, la correspondance n'est pas parfaite car cette nouvelle fonction se rapproche très fortement du mode de gestion de l'interruption BIOS 13h, avec des notions de cylindres, têtes et secteurs. Elle ne permet plus de balayer tous les secteurs en incrémentant simplement le numéro de secteur. Il est nécessaire de connaître le nombre de secteurs par piste, le nombre de têtes et le nombre de cylindres (par l'Int 21h/440Dh/40h) puis d'incrémenter et de tester chaque valeur ! Elle permet cependant d'accéder à des secteurs auparavant inaccessibles sur un disque dur (ceux de la piste zéro), au prix d'une correspondance imparfaite dans les numéros de secteurs.

La lecture par l'Int 21h/440Dh/61h s'effectue comme suit :

```
info  db 0              ; fonctions spéciales. Doit être à zéro.
      dw ?              ; numéro de tête.
      dw ?              ; numéro de cylindre.
      dw  ?             ; numéro du secteur sur la piste.
      dw  ?             ; nombre de secteurs à lire.
      dd ?              ; adresse complète du buffer.
...
mov   ax,440Dh          ; numéro de la fonction.
mov   cx,0861h          ; numéro de la sous-fonction.
mov   bx,drive          ; drive (0=défaut, 1=A, etc).
mov   dx,offset info    ; DS:DX pointe sur un bloc
                        ;  d'information.
int   21h               ; interruption DOS.
jc    error             ; traitement d'erreur si Carry.
```

INT 26h (Absolute Disk Write)

Permet d'écrire un ou plusieurs secteurs logiques à partir du drive spécifié. C'est le pendant exact de l'interruption 25h pour l'écriture. La documentation du DOS 5.0 recommande l'utilisation de la fonction de l'Int 21h/440Dh/41h (Write Track on Logical Drive), existante depuis le DOS 3.2, en lieu et place de l'interruption 26h. Les mêmes limitations s'appliquent à cette fonction.

INTERRUPTIONS NOUVELLEMENT DOCUMENTÉES

INT 28h (MS-DOS Idle Handler)

Cette interruption est appelée par certaines des douze premières fonctions de l'Int 21h (01h à 0Ch) qui attendent un événement pour se poursuivre (cas d'une entrée clavier, par exemple). Le traitement de l'interruption par le DOS ne consiste qu'en une instruction IRET. Par contre, un programme TSR peut détourner cette interruption pour savoir quand le DOS est en attente. Il lui est alors possible d'utiliser toutes les fonctions de l'Int 21h supérieures à 0Dh, même si l'indicateur "InDOS" (cf Int 21h/34h) n'est pas nul. L'interruption 28h a été utilisée dès la version 2.0 du MS-DOS.

INT 2Fh (Multiplex Interrupt)

Cette interruption est un point d'entrée commun à l'ensemble des TSR et device drivers du DOS, avec lesquels il devient possible d'échanger des informations. Elle a été "inaugurée" dès la version 2.0 par l'utilitaire PRINT du DOS et s'est agrandie au fur et à mesure des besoins. Elle est indexée avec un numéro de fonction dans AX et parfois d'autres paramètres. Lorsque le TSR correspondant n'est pas installé, l'interruption retourne avec le CARRY positionné et le registre AX égal à 1. Les différentes fonctions sont les suivantes :

* Int 2Fh/0100h (Get PRINT.EXE Installed State) permet de savoir si l'utilitaire PRINT.EXE est résident en mémoire.

```
mov   ax,0100h          ; numéro de fonction.
int   2Fh               ; interruption DOS Multiplex.
                        ; AL = 0FFh si PRINT est installé.
                        ; AL = 000h si PRINT non installé.
```

* Int 2Fh/0101h (Add File to Queue) permet d'ajouter un fichier dans la liste d'attente de PRINT.

```
mov   ax,0101h          ; numéro de fonction.
```

```
mov   dx,offset info    ; DS:DX pointe sur un bloc
                        ; d'informations.
                        ; composé d'un octet à 0, suivi
                        ; de l'adresse complète
                        ; (segment:offset) du nom du fichier
                        ; à ajouter (terminé par un 0).
int   2Fh               ; interruption DOS Multiplex.
jc    erreur            ; CARRY positionné en cas d'erreur.
```

* Int 2Fh/0102h (Remove File from Print Queue) permet d'enlever un fichier de la liste d'attente du PRINT.

```
mov   ax,0102h          ; numéro de fonction.
mov   dx,offset nom     ; DS:DX pointe sur le nom du fichier,
                        ; terminé par un 0.
int   2Fh               ; interruption DOS Multiplex.
jc    erreur            ; CARRY positionné en cas d'erreur.
```

* Int 2Fh/0103h (Cancel All Files in Print Queue) arrête l'édition en cours et enlève tous les fichiers de la liste d'attente.

```
mov   ax,0103h          ; numéro de fonction.
int   2Fh               ; interruption DOS Multiplex.
```

* Int 2Fh/0104h (Hold Print Jobs and Get Status) suspend l'édition en cours et renvoie l'adresse de la liste d'attente.

```
mov   ax,0104h          ; numéro de fonction.
int   2Fh               ; interruption DOS Multiplex.
                        ; DX contient le nombre d'erreurs
                        ; durant l'impression.
                        ; DS:SI pointe sur la liste d'attente.
```

* Int 2Fh/0105h (Release Print Jobs) permet de redémarrer une impression suspendue par la fonction précédente.

```
mov   ax,0105h          ; numéro de fonction.
int   2Fh               ; interruption DOS Multiplex.
```

* Int 2Fh/0106h (Get Printer Device) permet de retrouver l'adresse du device driver pour l'imprimante courante.

```
mov   ax,0106h          ; numéro de fonction.
int   2Fh               ; interruption DOS Multiplex.
jnc   liste_vide        ; liste d'attente vide
                        ; sinon DS:SI pointe sur le device
                        ; driver.
```

* Int 2Fh/0600h (Get ASSIGN.COM Installed State) permet de savoir si ASSIGN.COM a été chargé. AL vaut 0FFh en cas d'installation, 0 dans le cas contraire.

* Int 2Fh/1000h (Get SHARE.EXE Installed State) permet de savoir si SHARE.EXE a été chargé. AL vaut 0FFh en cas d'installation, 0 dans le cas contraire.

* Int 2Fh/1100h (Get Netword Installed State) permet de savoir si le redirecteur réseau a été chargé. AL vaut 0FFh en cas d'installation, 0 dans le cas contraire.

* Int 2Fh/1400h (Get NLSFUNC.EXE Installed State) permet de savoir si NLSFUNC.EXE a été chargé. AL vaut 0FFh en cas d'installation, 0 dans le cas contraire.

* Int 2Fh/1680h (MS-DOS Idle Call) permet d'informer le système ou les autres programmes, qu'un processus est en attente (entrée au clavier par exemple).

* Int 2Fh/1A00h (Get ANSI.SYS Installed State) permet de savoir si ANSI.SYS a été chargé. AL vaut 0FFH en cas de chargement, 0 dans le cas contraire.

* Int 2Fh/4300h (Get HIMEM.SYS Installed State) permet de savoir si HIMEM.SYS a été chargé. AL vaut 080H en cas de chargement, 0 dans le cas contraire.

* Int 2Fh/4310h (Get HIMEM.SYS Entry-Point Adress) permet de récupérer dans ES:BX l'adresse du point d'entrée du driver HIMEM.SYS. Ce point d'entrée permet d'envoyer des commandes au driver, selon la spécification XMS 2.0. Cependant, la communication avec le driver HIMEM.SYS doit être laissée au MS-DOS.

* Int 2Fh/4800h (Get DOSKEY.COM Installed State) permet de savoir si DOSKEY.COM a été chargé. AL est différent de 0 en cas de chargement.

* Int 2Fh/4810h (Read Command Line) permet de lire une ligne de commande et de la copier dans un buffer pointé par DS:DX. Lors de la lecture, les fonctions du DOSKEY ainsi que ses MACROS sont valides.

* Int 2Fh/4B01h (Build Notification Chain)
Int 2Fh/4B02h (Detect Switcher)
Int 2Fh/4B03h (Allocate Switcher ID)
Int 2Fh/4B04h (Free Switcher ID)
Int 2Fh/4B05h (Identify Instance Data) sont utilisées par le switcher de tâches incorporé au DOSSHELL de la version 5.0.

* Int 2Fh/AD80h (Get KEYB.COM Version Number) permet de connaître le numéro de version du driver clavier. Le numéro est rendu dans BX, avec la version majeure dans BH et la version mineure dans BL. Si la version est nulle, l'utilitaire KEYB.COM n'a pas été chargé.

* Int 2Fh/AD81h (Set KEYB.COM Active Code Page) permet de modifier le code page du driver KEYB.COM. La nouvelle valeur est donnée dans le registre BX. Le flag CARRY est

BIBLIOGRAPHIE

- Microsoft MS-DOS User's Guide and Reference.
- Microsoft MS-DOS Programmer's Reference (Microsoft Press).
- The MS-DOS Encyclopedia par Ray Duncan (Microsoft Press).
- Undocumented DOS par Addison Wesley.
- La Bible PC, programmation système (Editions Micro-Applications).
- Microsoft eXtended Memory Specification, Version 2.0.
- Microsoft MS-DOS 5 Evaluation Guide and Technical Notes.
- Microsoft Systems Journal, Juillet 91.

positionné si ce codepage n'est pas valide.

* Int 2Fh/AD82h (Set KEYB.COM Country Flag) permet de modifier le country du clavier.

```
mov   ax,0AD82h      ; numéro de fonction.
mov   bl,country     ; 00h pour clavier américain QWERTY
                     ; FFh pour clavier national.
int   2Fh            ; interruption DOS Multiplex.
jc    erreur         ; en cas d'erreur.
```

Cette fonction a le même résultat que l'appui de Ctrl-Alt-F1 ou Ctrl-Alt-F2 sur le clavier.

* Int 2Fh/AD83h (Get KEYB.COM Country Flag) permet de connaître l'état du clavier (QWERTY ou local). Le résultat est rendu dans BL, qui vaut 00h pour un clavier américain et FFh pour un clavier national.

* Int 2Fh/B000h (Get GRAFTABL.COM Installed State) permet de savoir si GRAFTABL.COM a été chargé. AL vaut 0FFh en cas d'installation, 0 dans le cas contraire.

* Int 2Fh/B700h (Get APPEND.EXE Installed State) permet de savoir si APPEND.EXE a été chargé. AL vaut 0FFh en cas d'installation, 0 dans le cas contraire.

* Int 2Fh/B702h (Get APPEND.EXE Version) récupère la version de l'utilitaire. AX vaut 0FFFFh pour les versions compatibles avec le DOS 5.0.

* Int 2Fh/B704h (Get APPEND.EXE Directory List Address) retourne dans ES:DI l'adresse de la liste des répertoires qui ont fait l'objet de la commande APPEND.

* Int 2Fh/B706h (Get APPEND.EXE Modes Flag) retourne dans le registre BX le mode d'opération courant de APPEND. Les différents modes possibles peuvent être une combinaison des valeurs suivantes :

0001h : APPEND autorisé.
1000h : APPEND s'applique aux opérations spécifiant un drive.
2000h : APPEND s'applique aux opérations spécifiant un chemin. C'est l'équivalent du switch /path:on lors du lancement de l'utilitaire.
4000h : APPEND stocke les répertoires ajoutés dans une variable d'environnement. C'est l'équivalent du switch /e lors du lancement de l'utilitaire.
8000h : APPEND s'applique aux opérations de chargement et de recherche de fichiers. C'est l'équivalent du switch /x:on lors du lancement de l'utilitaire.

* Int 2Fh/B707h (Set APPEND.EXE Modes Flag) permet de modifier le mode d'opération de l'utilitaire APPEND. Les flags sont spécifiés dans BX et peuvent être une combinaison des valeurs listées dans la fonction précédente.

* Int 2Fh/B711h (Set True-Name Flag) demande à l'utilitaire APPEND de modifier le nom du fichier pour y inclure le répertoire, et ce pour la prochaine fonction 3Dh (Open File with Handle), 4300h (Get File Attributes), 4Eh (Find First File) et 6Ch (Extended Open/Create).

* Int 2Fh/Cxxxh à Int 2Fh/FFFFh sont réservées pour les programmes et TSR d'application. Cette solution peut remplacer avantageusement tout autre moyen de communication entre deux programmes dont l'un doit être résident. Elle permet de tester de façon simple l'installation en résident, et surtout, elle a été implémentée dans le DOS dès la version 2.0. En règle générale, la valeur du code dans AL indique la fonction à réaliser, tandis que la valeur de AH est représentative du programme. En raison du peu de code disponible (64 de C0h à FFh), il semble toutefois souhaitable de prévoir la modification de ce code à partir de la ligne de commande (par exemple, sous la forme : commande /multiplex:code).

INTERRUPTIONS SUPPRIMÉES

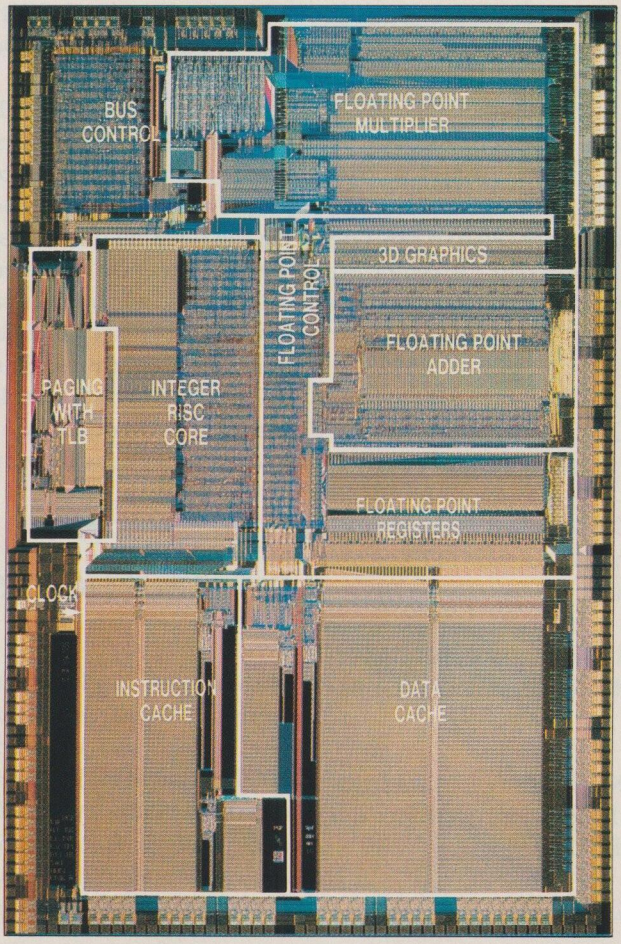
Plusieurs interruptions autrefois utilisées par le DOS ont été supprimées à partir de la version 5.0. Bien sûr, rien d'important, puisque ces interruptions datent toutes du DOS 1.0, qu'elles étaient exclusivement présentes pour des problèmes de compatibilité avec CPM/80 et qu'elles n'ont jamais été documentées. Autant dire qu'il est peu probable qu'elles aient jamais été utilisées dans un programme d'application, et que leur disparition ne cause aucune incompatibilité. □

LES COPROCESSEURS ARITHMÉTIQUES DU 386 DX ET DU i486

ATTA OLOUMI

Les coprocesseurs ont un objectif commun : accélérer le calcul, avec, comme pierre angulaire, la compatibilité avec l'Intel i387. Autour de cette variation de thème, plusieurs acteurs : Cyrix, IIT, Intel et Weitek. Avec l'intégration du coprocesseur arithmétique, Intel pose le i486 comme la nouvelle référence à dépasser. Cette tâche est d'autant plus ardue que l'intégration sur une même puce de l'unité centrale et de l'unité de traitement à virgule flottante double les performances grâce à la diminution du temps de transfert des données entre les deux composants.

Cependant, il ne faut pas laisser les benchmarks trop généralistes occulter les capacités spécifiques des produits concurrents : le IIT 3C87 pourrait bien donner du fil à retordre à ses compétiteurs dans certaines applications graphiques de type 3D, le Cyrix 83D87 se distingue par sa précision et sa rapidité lors des calculs trigonométriques, les Weiteks pour leur mode de communication avec le processeur principal, et l'Intel i860, utilisé comme coprocesseur arithmétique, offre en théorie une puissance crête qui demeurerait inégalée. Un coprocesseur est un élément indispensable pour éviter que les performances de l'unité centrale ne s'écroulent dans l'émulation des opérations arithmétiques en virgule flottante. Cette faiblesse est la conséquence de la structure des registres de l'unité centrale qui travaillent avec des entiers (par exemple, avec des entiers définis sur 32 bits dans le cas du i386). Le coprocesseur arithmétique s'avère donc nécessaire dès que les variables d'un progamme sont déclarées comme étant des réels : on aborde là le domaine du calcul flottant, quelle que soit l'application en cours (calcul de TVA dans un logiciel de comptabilité, rotation d'une image avant son placement dans un logiciel de PAO, etc.)



Le 387 DX

Le 387 DX apporte un complément d'instructions arithmétiques à l'unité centrale du 386 DX. Il le décharge du traitement en virgule flottante, lui permettant ainsi de s'occuper d'autres tâches telles que la recherche d'adresse. Autrement dit, le 387 DX effectue non seulement des calculs complexes pour le 386 DX, sans pour autant interrompre le travail de ce dernier. L'architecture interne du 387 DX est sur 80 bits. Les données sont transmises par l'intermédiaire d'un bus externe 32 bits, et sont ensuite converties suivant leur type réel (virgule flottante sur 32, 64 ou 80 bits).

Le cœur du coprocesseur est constitué de 8 registres sur 80 bits, utilisables individuellement comme registres généraux. Ils sont adressés par deux bus à la fois : un bus de 16 bits pour la partie exposant et un bus sur 68 bits pour la mantisse. Chacun d'eux est relié à des unités opératoires telles que des unités d'addition ou de décalage. Les deux constituants d'un réel (l'exposant et la mantisse)

En intégrant une unité de traitement mathématique dans le processeur i486, Intel met en avant une solution haut de gamme destinée aux utilisateurs de calculs en virgule flottante, généralement des scientifiques ou des utilisateurs de logiciel de CAO. Les coprocesseurs séparés deviennent-ils pour autant obsolètes ? Ce serait négliger les spécificités des différents coprocesseurs qui n'attendent que le bon vouloir des développeurs pour être exploités par les utilisateurs.

sont donc traités parallèlement par des ensembles séparés. Le choix d'une architecture sur 80 bits est judicieux dans la mesure où le coprocesseur peut aborder n'importe quelle précision de travail. Mais lors du traitement courant en simple précision, cette architecture n'offre aucun gain car les registres demeurent à moitié inemployés.

A noter également qu'aucune opération n'est câblée, c'est-à-dire qu'elles ne sont pas implantées physiquement sur le silicium, contrairement à d'autres coprocesseurs, tels que les Cyrix et les Weiteks, dans lesquels multiplication et addition sont câblées et s'effectuent en une seule passe.

Enfin, le 387 DX comporte une mémoire morte contenant le microcode destiné à générer les instructions arithmétiques, trigonométriques, exponentielles et logarithmiques ainsi que les instructions de comparaison. Ce composant serait satisfaisant s'il ne souffrait d'une lacune majeure liée à son mode de communication avec l'unité centrale : celle-ci adresse les instructions et les données par l'intermédiaire d'un bus unique, à savoir le bus de données (sur 32 bits).

Il se forme donc à l'entrée du coprocesseur un fâcheux goulot d'étranglement. Avant de commencer une opération, plusieurs temps de cycle sont utilisés pour charger les données et les instructions, qui sont ensuite décodées et remplacées par la séquence microcode associée. Comme aucune instruction ne peut être effectuée en un ou deux cycles d'horloge, ce temps de calcul pourrait être mis à profit pour préparer les instructions suivantes.

Malheureusement, le 387 ne dispose que d'une mémoire intermédiaire constituée de 2,5 registres de 32 bits. Cette valeur surprenante correspond à l'équivalent d'un seul registre sur 80 bits. Il n'y a donc pas de solution de type mémoire cache au problème du goulot d'étranglement.

En réponse à ces critiques, Intel renvoie à l'usage réel qui est fait du coprocesseur 387 : la grande majorité des utilisateurs ne sont pas consommateurs intensifs de calculs flottants. Autrement dit, le gain apporté par le 387 est suffisant dans les cas courants. Lorsque les besoins en puissance de calcul deviennent importants, il faut alors s'orienter vers le i486.

Figure 1 :
Illustration des flots de données entre ALU et FPU sur 386/387 et 486

Formats de données	Grandeur associée	Précision	Nbre de bits de signe	Nbre de bits d'exposant	Nbre de bits de mantisse
Entier	10^4	16	0	0	16
Entier court	10^{12}	32	0	0	32
Entier long	10^{19}	64	0	0	64
Précision simple (32 bits)	$10^{-/+38}$	24	1	7	24
Précision double (64bits)	$10^{-/+308}$	53	1	10	53
Précision étendue (80 bits)	$10^{-/+4932}$	64	1	15	64

L'i486 : un couple 386/387 optimisé

Plutôt qu'un nouveau processeur, l'i486 doit être considéré comme l'aboutissement de la famille 386. Par rapport au couple 386/387, il ne contient que six instructions supplémentaires : trois pour la gestion des 8 Ko de mémoire cache intégrés, et trois autres destinées à en faciliter l'usage dans un environnement multiprocesseur. Ces principales avancées tiennent en trois points :

- optimisation des instructions les plus couramment utilisées afin de les rendre exécutables en un minimum de cycles, voire en un seul (en cela la frontière entre RISC et CISC s'estompe, sans pour autant renoncer à la compatibilité descendante)
- intégration d'une mémoire cache de 8 Ko

LES COPROCESSEURS ARITHMÉTIQUES DU 386 DX ET DU I486

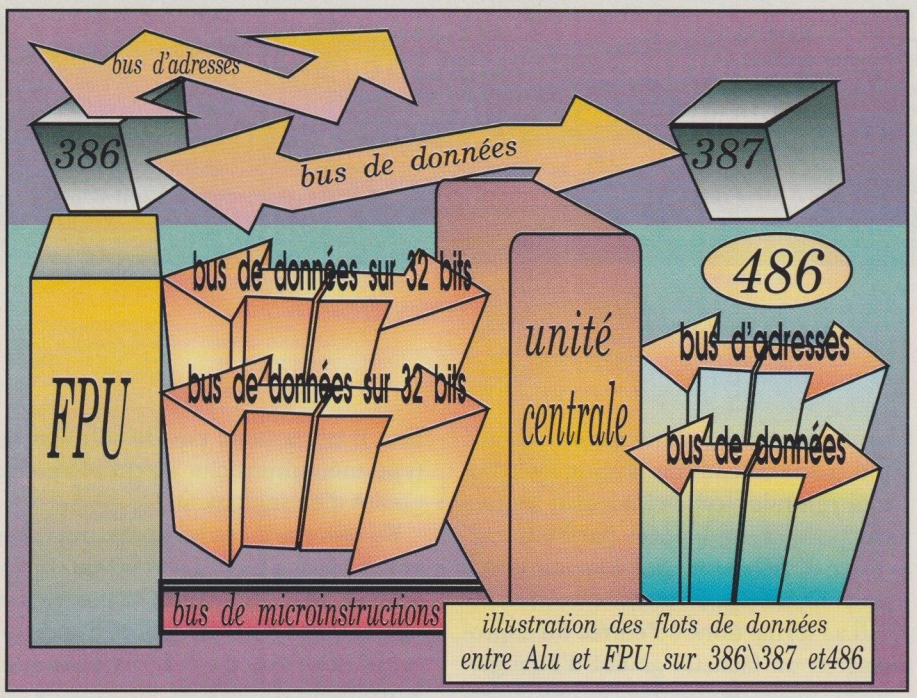


illustration des flots de données entre Alu et FPU sur 386\387 et486

- intégration au sein d'un seul composant du processeur et du coprocesseur arithmétique.

L'intégration du coprocesseur a donné l'occasion de multiplier les bus y donnant accès. L'unité de décodage des instructions est commune à la partie entière et à l'unité à virgule flottante (FPU ou Floating Point Unit) au travers d'un bus spécifique (*cf figure 2*). Si le microcode a été optimisé, l'architecture du FPU n'a pas varié par rapport au 387 DX. Les gains découlent donc essentiellement de l'amélioration de l'interfaçage entre les deux composants. Selon Intel, il en résulterait une diminution d'un facteur 2 à 4 du nombre de cycles d'horloge nécessaires pour l'exécution des instructions à virgule flottante. Le modèle i486 ne doit pas être confondu avec le 486 SX.

Ce dernier est une version dégradée du i486, où le FPU a été inhibé. Il est présenté dans un boîtier quasiment identique (une seule patte change). Pour retrouver les fonctions de calcul en virgule flottante, il faut acquérir le 487 SX qui n'est rien d'autre qu'un i486 complet. Par conséquent, l'installation du 487 SX sur son support a pour effet de désactiver le 486 SX !

Les compatibles

Si le i486 apporte une réponse à l'attente des utilisateurs en quête de puissance de calcul, ce n'est pas pour autant l'unique solution. D'autres produits constituent en effet des alternatives au changement de processeur principal. Le Cyrix FastMath 83D87 joue la carte de la compatibilité avec le 387 DX, à tous les niveaux (boîtier et microcode). Il comprend donc un ensemble de 8 registres sur 80 bits.

La consommation énergétique constitue l'une différences notables avec le 387 DX. Celle du Fastmath est plus faible en raison de sa technologie CMOS. Par ailleurs, Cyrix annonce un surprenant gain d'un facteur 3 à 10 dans le nombre de cycles nécessaire à l'exécution de certaines instructions. Afin d'y parvenir, le 83D87 exploite le câblage des fonctions : le composant d'Intel perd plusieurs cycles à remplacer l'instruction multiplication par le microcode associé, puis à dérouler chaque étape de ce dernier. Le composant de Cyrix ne nécessite pour sa part qu'une étape de chargement des registres de l'unité de calcul, la multiplication proprement dite étant câblée. Quant à l'opération d'addition, elle est accélérée par un circuit dédié. Mais cette fonction n'est pas câblée : il n'existe pas d'unité «additionneur» implémentée dans cette puce. Les circuits dédiés à l'addition et à la multiplication accélèrent également les instructions de division, soustraction et racine carrée, qui sont microcodées et sont basées sur les précédentes. Mais l'originalité principale du 83D87 réside dans les fonctions transcendantes (sinus, cosinus, tangente, exponentiel, etc.).

Mathématiquement, l'obtention d'une fonction transcendante passe par une approximation polynômiale, c'est-à-dire que la fonction est remplacée par un polynôme dont le résultat est très proche de la fonction initiale. Les coprocesseurs d'Intel utilisent l'algorithme CORDIC qui évite un nombre important de calculs intermédiaires en n'utilisant que des additions et des décalages.

Figure 3 :
Performances comparées du 387 et du 486

	Nombre de cycles		
Instruction	8088/8087	80386/80387	8046
Addition en virgule flottante sur 80 bits	70-100	23-31	8-20
Multiplication en virgule flottante sur 80 bits	130-145	29-57	16
Division en virgule flottante sur 80 bits	193-208	88	73

Remarque : à titre de comparaison, on notera que le Cyrix 83D87 utilise 6 cycles pour effectuer une addition en double précision (64 bits) lorsque les opérandes sont dans les registres internes, contre 15 cycles lorsqu'elles sont en mémoire.

Les ingénieurs de Cyrix n'ont guère été satisfaits de la précision obtenue avec cet algorithme. Mais leur solution d'algorithme polynômial pose deux problèmes : il entraîne d'une part une inflation des opérations d'addition et de multiplication (et ce d'autant plus que la précision désirée est élevée), et il multiplie d'autre part les résultats intermédiaires qui ne peuvent pas se satisfaire des 8 registres de travail principaux.

En réponse à ces deux problèmes, la solution de Cyrix consiste à utiliser, d'une part, la rapidité de la fonction multiplication câblée et, d'autre part, des registres spécifiques supplémentaires sur 90 bits afin d'y stocker les résultats intermédiaires (les dix bits complémentaires étant destinés à compenser les erreurs de calcul dues aux troncatures). Ces registres particuliers ne sont pas accessibles aux développeurs. Au final, le i387 est surclassé par le coprocesseur de Cyrix sur deux points : en précision pour les fonctions transcendantes, et en rapidité. Il n'en va pas de même avec le i486 qui conserve l'avantage, au niveau de la vitesse de calcul tout au moins, grâce à la disparition du goulot d'étranglement à l'entrée du FPU.

Les composants de Cyrix ont en effet hérité, avec la compatibilité totale du i387 DX, des problèmes de transfert de données. L'approximation polynômiale n'est pas l'exclusivité des coprocesseurs de Cyrix puisqu'elle est reprise par IIT pour son 3C87 (C pour CMOS). Il intègre également des câbles, mais il travaille à une précision moindre que le Cyrix et son niveau de performances est légèrement en deçà de celui du FastMath.

L'IIT contient 24 registres de travail, au lieu de 8 pour le i387, qui sont organisés en trois groupes de huit registres.

CALCUL FLOTTANT

Une variable peut être définie comme étant du type entier ou réel. Le calcul avec des entiers de 8 à 32 bits est réalisable au sein même de l'unité centrale, y compris les opérations mathématiques sur les entiers, sans que les performances du microprocesseur ne se dégradent notablement. Par contre, la moindre addition ou multiplication en virgule flottante réclame des dizaines d'instructions, car le processeur ne comprend qu'une unité de traitement d'entier (appelée ALU), et doit émuler les instructions en virgule flottante. Le coprocesseur arithmétique est l'unité spécifiquement destinée au traitement des données à virgule flottante.

A cette fin, il contient des registres de travail en précision simple, double ou étendue, ainsi que des instructions et des microcodes spécifiques. Une norme internationale de la représentation des données en calcul flottant a été définie en 1985 : il s'agit du standard ANSI/IEEE 754-1985 (cf figure 1). La conformité à cette norme permet de conserver l'investissement des développements. Elle est reprise par le coprocesseur Intel i387 qui est devenu à son tour une norme.

Trois nouvelles instructions désignent lequel de ces trois groupes doit être utilisé pour les opérations en cours. Convenablement gérés par le développeur, les registres supplémentaires du 3C87 apportent un gain qui peut être notable du fait de la diminution des échanges de données avec le 386 DX.

Encore faut-il renoncer à la compatibilité logicielle avec le 387... Une quatrième instruction, F4X4, tire réellement parti de l'organisation interne. Elle multiplie deux matrices 4X4 entre elles. Pour ce faire, elle utilise l'ensemble des registres disponibles et ne réclame pas d'échange avec le processeur principal pour stocker les données intermédiaires. Sur cette opération spécifique, qui engendre plusieurs dizaines de multiplications et additions, le coprocesseur d'IIT est sans concurrent.

Comme la gestion des flots de données doit être optimale lors du remplissage des registres, il est préférable de s'en remettre à une programmation en assembleur. Cette fonction spéciale est la plus utilisée dans les algorithmes de déplacement autour d'un graphique 3D. Sachant que le graphisme tridimensionnel est l'un des principaux consommateurs de calculs à virgule flottante, on aprécie le pragmatisme des ingénieurs de IIT. Mais l'assembleur n'étant plus dans l'air du temps, cette originalité reste inexploitée, malgré quelques déclarations d'intention.

Au-delà de la compatibilité

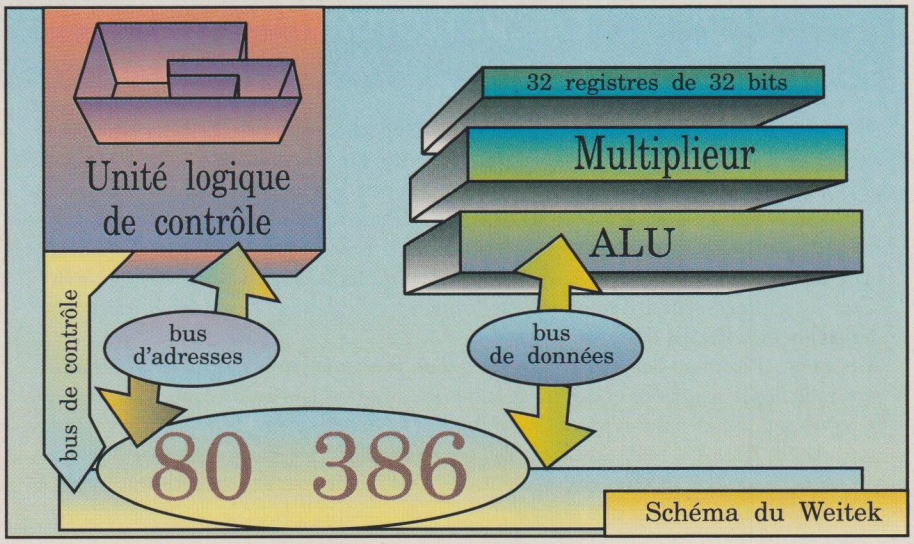
Il est difficile de faire mieux sans se débarrasser des contraintes liées aux transferts de données et imposées par l'architecture du couple 386/387. C'est la voie choisie par Weitek pour ses coprocesseurs Abacus 3167 et 4167. Abandonnant résolument la compatibilité avec la famille 80X87, ils diffèrent par leur architecture interne, leurs instructions et leur support boîtier. Contrairement aux huit registres de travail du 387, les Weitek sont pourvus de 32 registres sur 32 bits. Ces derniers peuvent être regroupés pour former un groupe de 16 registres de 64 bits.

Le multiplieur câblé et l'ALU exécutent les opérations en simple précision en un cycle d'horloge. Estimant qu'ils ne pouvaient apporter un gain significatif sans reprendre les concepts de la communication processeur/coprocesseur, les ingénieurs de Weitek ont adopté la technique du «memory map».

Figure 4 : Constitution d'une adresse destinée au Weitek.

Champs de sélection du coprocesseur		Code opératoire		Adresses des opérandes	
31	16	15	10	9	0

LES COPROCESSEURS ARITHMÉTIQUES DU 386 DX ET DU I486



Cette dernière consiste à utiliser à la fois le bus de données et le bus d'adresse du 386 DX.

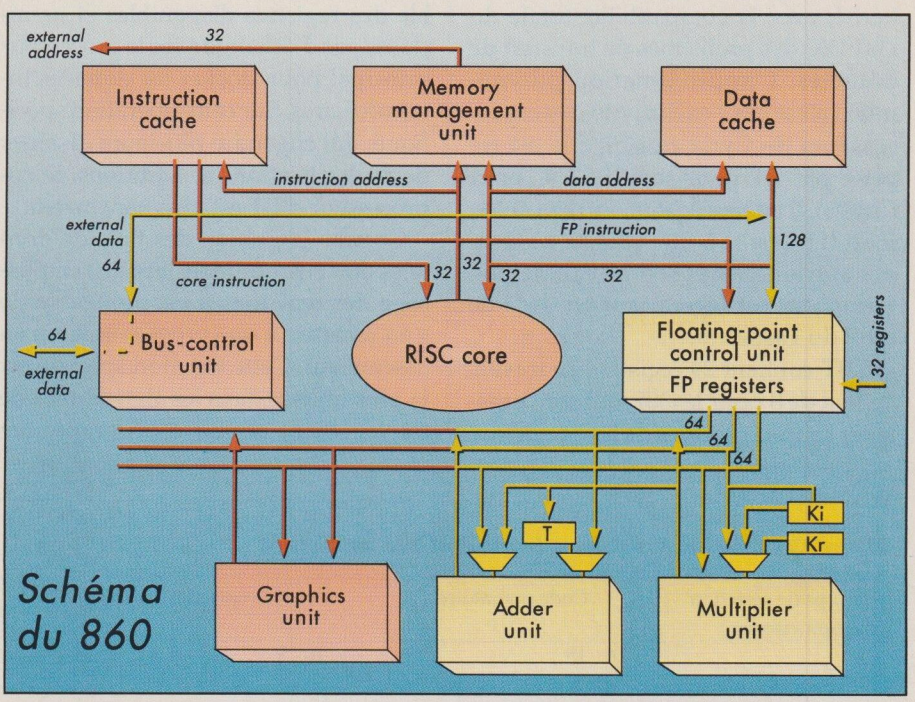
Les instructions sont transmises par le canal du bus d'adresse et les opérandes par le bus de données. L'originalité du procédé réside dans l'usage du bus d'adresse : pour l'unité centrale, le Weitek est considéré comme un espace mémoire physique de 64 Ko. Cette valeur, sur 16 bits, est située entre l'adresse C000:0000h et l'adresse C000:FFFFh. Chaque adresse commençant par la valeur hexadécimale C000 est par conséquent reconnue par le Weitek comme étant l'indication d'une instruction.

Celle-ci est spécifiée dans les six bits suivants de l'adresse, tandis que les 10 bits restants servent à localiser les opérants. Les liaisons avec le processeur sont dès lors pleinement opérationnelles, le goulot d'étranglement ayant disparu. Encore faut-il veiller à ne pas perdre l'acquis par une mauvaise programmation. La spécificité de la programmation réside dans la gestion de la mémoire de travail disponible en interne, c'est-à-dire de l'ensemble des 32 registres. Dans la pratique, il faut veiller à minimiser et à regrouper les instructions dont les opérandes sont situées en mémoire vive.

Remarquons cependant que la plus-value d'une programmation en assembleur pour ce composant est limitée, l'espace de travail interne étant réduit. Le doublement de la largeur des voies de communication accélère effectivement les applications traitées à l'aide d'un compilateur compatible Weitek. Cependant, les fonctions transcendantes ne sont pas microcodées. D'après les responsables de la firme, ces fonctions sont présentes dans les librairies qu'ils proposent pour les compilateurs compatibles avec les Weitek (par exemple, les compilateurs NDP de Microway). Malheureusement, la multiplication des transferts d'informations qui en découle annule les acquis de l'architecture du composant. La gamme des coprocesseurs Weitek destinés à accompagner les 386 DX et i486 comprend le 3167 et le 4167. Le 3167 a remplacé le 1167. Bien que doté d'une structure commune, ce dernier est en fait un circuit imprimé contenant trois circuits dont un multiplieur et un additionneur, tandis que le 3167 est un cicuit intégré unique. Tous les deux se montent sur le support EMC (Extended Math Coprocessor), support également utilisé par un nouveau composant de Cyrix : le EMC87. L'intérieur de la puce est identique au 83D87, il est compatible avec le 387 DX mais peut également être adressé en mode «memory map». L'utilisation de ce mode réclame, comme pour l'Abacus, un compilateur spécifique.

Parallélisme et pipe-line

Au-delà de la puissance offerte par les coprocesseurs à opérations arithmétiques câblées, il faut utiliser les architectures issues des super-ordinateurs, no-



tamment celles qui permettent d'accomplir une instruction à virgule flottante ou plus par cycle (l'Abacus 3167, par exemple, en consomme 3 à 5). Tel est le cas de l'Intel i860, surnommé "Cray sur un chip". On le trouve sur des cartes mères de compatibles PC haut de gamme. La société américaine Hauppauge propose ainsi une carte mère au standard EISA à base d'un i486 et d'un i860, qui ont tous deux accès à une mémoire commune de 64 Mo.

Le i860 est composé d'un processeur RISC travaillant sur 32 bits entiers, de deux caches (qui dans la nouvelle version XP, atteignent 16 Ko chacun), d'un FPU constitué d'un ensemble de 32 registres de travail, d'un additionneur, d'un multiplieur et d'un Z-buffer (à l'intention des opérations graphiques 3D), le tout étant câblé sur 64 bits. Les bus de données sont également sur 64 bits, à l'exception du bus de liaison entre la mémoire cache des données et le FPU qui est sur 128 bits. L'unité de traitement d'entier sur 32 bits a une structure RISC classique. Avec le i486, la plupart des instructions équivalentes s'effectuent également en un cycle. Ce n'est que dans le rôle de coprocesseur arithmétique que l'i860 concède des performances inégalées. Celui-ci a en effet une puissance crête annoncée de 80 Mflops à 40 MHz, contre 3 ou 4 Mflops pour le i486. Ces chiffres appellent quelques commentaires.

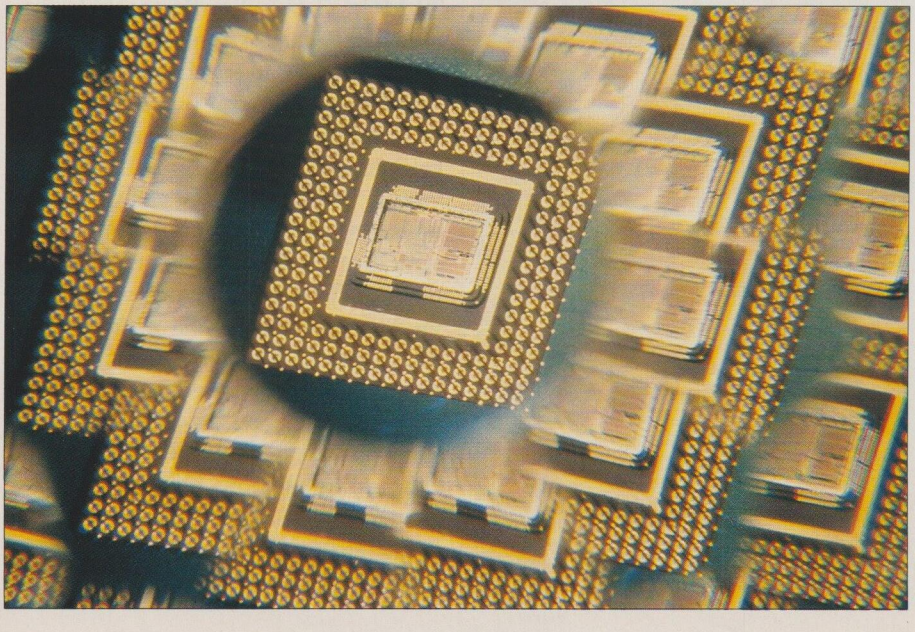
Figure 7 : Liste des instructions "pipe-line"

pf mul	multiplication en virgule flottante
pf mul3	multiplication en virgule flottante
pf add	addition en virgule flottante
pf sub	soustraction en virgule flottante
pf gt	
pf eq	opérations de comparaison
pf le	
pf ix	conversion de variables en virgule flottante en entier
pf trunc	troncature de variables en virgule flottante en entier
pf am	instruction double d'addition et de multiplication
pf sm	instruction double de soustraction et de multiplication
pf mam	instruction double de multiplication avec addition
pf msm	instruction double de multiplication avec soustraction

Le i860 opère en simple précision ou en double précision. Autrement dit, la précision étendue sur 80 bits doit être émulée par logiciel. Le travail en mode pipe-line n'est alors guère possible et il faut donc se contenter du mode scalaire, où une multiplication sur 64 bits réclame 4 cycles d'horloge. Une multiplication sur 80 bits nécessite au moins quatre fois plus de temps. Or, le i486 effectue dans le meilleur des cas une multiplication sur 80 bits en 16 cycles. L'intérêt du i860 est donc bien moindre qu'il n'y paraît dès que les variables sont définies sur un standard autre que ceux de la virgule flottante simple précision et de la virgule flottante double précision. La puissance du i860 réside dans la structure en «pipe-line» de l'additionneur et du multiplieur câblés. Le principe de cette architecture est analogue à celui d'une chaîne de fabrication de voitures où l'unité de montage des moteurs s'occupe d'une voiture tandis que l'atelier de soudure travaille sur la carrosserie d'une voiture précédente. Cet enchaînement d'étapes qui s'effectuent simultanément constitue le principe du pipe-line.

Ce qu'on appelle le nombre d'étages du pipe-line correspond dans notre exemple au nombre d'ateliers. L'additionneur et le multiplieur sont à trois étages. En simple précision, l'addition et la multiplication réclament trois opérations, de même que l'addition sur 64 bits. Dans les pipes-lines à trois étages, ces opérations s'effectuent donc en un unique temps de cycle.

En revanche, la multiplication sur 64 bits demande quatre opérations et ne peut donc se contenter d'un seul cycle d'horloge. Il est possible, en mode pipeline, d'effectuer deux opérations à virgule flottante à la fois (par exemple une multiplication et une addition), à l'aide des instructions doubles. Ceci explique la puissance crête de 80 Mflops, en simple précision, pour une fréquence



Le microprocesseur i860 d'Intel peut aussi être utilisé comme coprocesseur. En calcul en virgule flottante, il demeure sans concurrent.

d'horloge de 40 MHz. Autre organisation performante, le bus d'instructions sur 32 bits peut contenir une instruction destinée à la section RISC et une pour le FPU. Durant ce mode appelé "superscalaire", les opérations de recherche et d□e chargement ont lieu parallèlement aux calculs dans le FPU.
En dehors de dix instructions graphiques 3D (grâce auxquelles le i860 est utilisé comme coprocesseur dédié sur nombre de cartes graphiques), le FPU ne comprend que des instructions arithmétiques de base (addition, multiplication, soustraction) et des instructions de comparaison. Les outils de programmation doivent donc contenir une bibliothèque de fonctions mathématiques.

Cependant, les outils de programmation et de simulation de bas niveau d'Intel (simulateur, assembleur, éditeur de liens et debugger) ne sont disponibles que sous Unix (Unix System V/386). L'exploitation des potentialités du i860 passe obligatoirement par la programmation en assembleur, sinon le risque est grand de voir les originalités du composant disparaître : les compilateurs sont en effet peu aptes à gérer trois niveaux de parallélisme à la fois (instructions superscalaires, pipelinées et doubles). Le test de la carte Number Smasher 860 de Microway *(voir InfoPC numéro 68 de février 91)* montre un gain moyen de 460% par rapport à un PC 386/387 à 25 MHz. Si l'on tient compte du fait que la fréquence de travail du i860 est pratiquement le double et que le i486 est deux à quatre fois plus rapide que le couple 386/387 (à fréquence égale), le gain est limité à 30% par rapport au i486.

Ce résultat est confirmé, par ailleurs, par le Benchmark Doduc qui crédite le i860 (à 40 MHz) d'une vitesse relative de 276 contre 191 pour un i486 (à 33 MHz), la base 1 étant un 386SX à 16 MHz (sans coprocesseur). Avec le i860 comme coprocesseur, la frontière qui sépare la micro du monde des stations de travail s'estompe, tout au moins en ce qui concerne la puissance de calcul en virgule flottante. Pour mémoire, le haut de gamme des stations de travail est occupé par IBM avec le RS/6000 à 30 MHz et par Apollo (racheté par Hewlett-Packard) avec le HP/Apollo 9000 Series 700 (à 66 MHz).

Ces machines sont qualifiées à plus de 20 Mflops. Leur processeur RISC propriétaire est muni d'une unité de calcul à virgule flottante utilisant également des opérateurs arithmétiques câblés, une structure pipe-line et des instructions doubles. Deux voies demeurent pour gagner en performance : la multiplication du nombre de FPU, avec les difficultés de programmation inhérentes à la gestion du parallélisme, ou le câblage de fonctions spécifiques liées à certains algorithmes. Les stations Silicons Graphics sont l'exemple réussi d'une architecture définie en fonction d'un domaine d'application particulier (le graphisme 3D). □

Figure 8

Le mode pipe-line consiste a commencer la première étape d'une instruction avant que l'instruction précédente ait été complétement effectuée.
Soit la séquence :

```
pfmul x1,y1,z0
pfmul x2,y2,z0
pfmul x3,y3,z0
pfmul x4,y4,z1
pfmul x5,y5,z2
```

Elle correspond à la multiplication d'une pile xi et d'une yi, le résultat étant reporté sur la pile zi. En simple précision, l'instruction pfmul (multiplication "pipe line") requiert 3 étapes.

Analyse du déroulement par cycle d'horloge :

Cycle d'horloge	z1	z2	z3	z4
1	*	_	_	_
2	*	*	_	_
3	*	*	*	_
4	x1+y1	*	*	*
5	x1+y1	x2+y2	*	*
.....				

La même séquence avec des instructions scalaires :

```
mul x1,y1,z1
mul x2,y2,z2
mul x3,y2,z3
mul x4,y4,z4
.....
```

Le déroulement par cycle d'horloge est alors le suivant :

Cycle d'horloge	z1	z2	z3	z4
1	*	_	_	_
2	*	_	_	_
3	*	_	_	_
4	x1+y1	*	_	_
5	x1+y1	*	_	_
6	x1+y1	*	_	_
7	x1+y1	x2+y2	_	_
.....				

Donc, en mode pipe-line, une opération en virgule flottante peut être effectuée en un cycle.
(Le signe * représente un processus de calcul en cours)

CLARION Personnel

N°1 AUX ETATS-UNIS

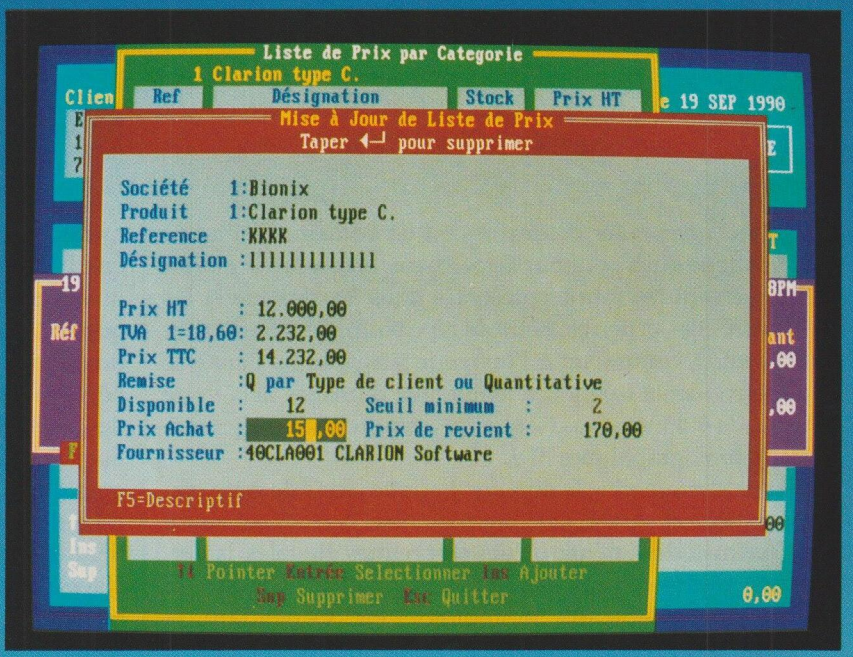
EN VERSION FRANÇAISE

Le générateur d'applications intégral

Le moyen le plus simple pour réaliser des applications de gestion de toute nature

Créez vos applications de A à Z en quelques minutes, c'est ce que CLARION Personnel le générateur le plus primé aux U.S.A. réalise pour vous. Fichiers, index, écrans multifenêtrés, menus, tableaux déroulants, masques de saisie, impression d'étiquettes, de factures, de lettres...

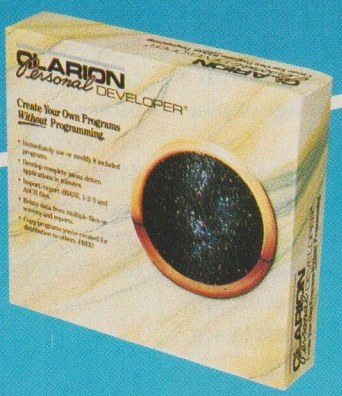
Pour des applications **SGBD** Relationnel de qualité, distribuable **sans redevance**.



Prix tarif 2390,00 Francs HT

Prix de lancement

990 F



EN CADEAU

8 applications types, déjà développées,utilisables telles quelles ou modifiables (Inventaire, Gestion de Club, de Bibliothèque, Mailing, Facturation, Répertoire téléphonique, Suivi commercial, Gestion scolaire).

Matériel requis : micro-ordinateur compatible PC et système d'exploitation MS-DOS, 512 K RAM, 2 MO sur disque.

Spécifications techniques : Multifichiers, 252 fichiers par application, 251 clés par fichier, 255 champs par enregistrement. Longueur enregistrement jusqu'à 64 Ko.

Taille des fichiers : Illimitée.

Mode automatique : Crée une application totale en quelques secondes (menus, tableaux déroulants, masques de saisie, états) à partir de la simple définition du fichier

Aide à l'utilisation : Intégrée (plus de 400 écrans).

Conversion de fichiers : Import/export avec fichiers dBase, DIF, ASCII délimité.

Pour plus d'informations, cerclez 200

Service lecteurs n° 7 (dernière page)

Bon de commande

Je désire recevoir CLARION Personnel
pour le prix de lancementde 990,00 Francs HT
plus la TVA et ma participation aux frais d'envoi soit 1 249,14 Francs TTC

Nom .....
Société .....
Fonction .....
Adresse .....
Code postal |_|_|_|_|_| Ville .....
Tél .....

Signature

Veuillez trouver ci-joint mon chèque de règlement à l'ordre de Bionix.

BIONIX SOFTWARE

32, rue Molitor
75016 Paris - FRANCE
Tél. : (33)-(1) 40 71 09 36 - Fax. : (33)-(1) 46 51 24 26

Pour la SUISSE
IHS DISTRIBUTION S.A.
En Budron A 16
1052 Le mont / Lausanne (Suisse)
Tél. : (021) 652 34 36 - Fax. : (021) 652 34 52

Pour la BELGIQUE
NASA INFORMATIQUE S.A.
Rue Godefroid, 30
5000 Namur
Tél : (081) 22 40 77 - Fax : (081) 23 19 35

THÉORIE

LA DÉTECTION DES CONTOURS DANS LES IMAGES

CHRISTOPHE SOUCHARD

La reconnaissance des formes est un domaine de l'informatique qui a pour but de percevoir et reconnaître automatiquement les formes présentes dans les signaux et les images. Celles-ci peuvent être de type bidimensionnel, tridimensionnel, comme pour l'imagerie sismique ou l'analyse de mouvements (séquence temporelle d'images 2D), voire même de 4 dimensions (séquence temporelle d'images médicales tomographiques 3D).

On ne compte plus le nombre d'applications de cette discipline riche et variée : elles ont pour objet la reconnaissance de l'écriture, de la parole ou des empreintes digitales, la perception tridimensionnelle pour la navigation de robot, les diagnostics médicaux, la géologie, etc. Cependant, la complexité des formes naturelles constitue toujours un obstacle majeur pour la reconnaissance, et souvent il n'est pas possible d'effectuer les traitements nécessaires directement sur les données de départ. C'est pourquoi l'on s'efforce généralement d'obtenir une description compacte et structurée des images, à l'aide de primitives simples, davantage exploitable qu'un ensemble de pixels.

Ainsi le processus de reconnaissance des formes dans les images est-il classiquement scindé en deux phases :

- extraction d'attributs caractérisant les entités de l'image. Les attributs étudiés correspondent le plus souvent à des points d'intérêt ou à certaines zones de l'image et en particulier aux contours et aux régions. Cette étape est la segmentation.
- traitement des caractéristiques extraites. Il s'agit de la reconnaissance proprement dite.

La représentation de l'image générée par la segmentation conditionne tous les traitements ultérieurs. De nombreux travaux lui ont été consacrés (cf bibliographie en fin d'article). Mais la diversité des signaux, la difficulté du problème, la prodigieuse évolution des ordinateurs, en particulier l'arrivée des réseaux neuronaux, ainsi que l'origine variée des chercheurs qui ont conduit ces travaux, ont mené à des algorithmes et méthodes très variés qui ne donnent pas tous les mêmes résultats. Ainsi, les détecteurs de contours sur les images se bousculent : Gradient, Laplacien, Prewitt, Sobel, Kirsh, Nagao, Canny, Deriche, etc. A partir d'une telle représentation par contour, on peut extraire des primitives (segments, arcs de cercle,...) par des algorithmes classiques tels que l'algorithme de la corde, et leur donner suivant le cas une structure de chaîne, d'arbre ou de graphe général. Quelle que soit l'origine de l'image, elle constitue une représentation d'un univers composé d'entités : les objets, qui sont généralement des composantes connexes de l'image. L'étude des objets constitutif de l'image nécessite l'extraction d'attributs divers (topologie, texture, ...) sur les zones caractéristiques que sont les contours et les régions.

La difficulté de la détection des contours ou des régions tient essentiellement au fait que l'on se heurte dans cette étape à un obstacle de taille, à savoir la réalité de l'image, qui est une matrice de valeurs issues d'un mélange de phénomènes physiques et géométriques complexes. C'est ainsi, par exemple, que le bruit présent dans les images (bruit d'échantillonnage, bruit du capteur, ...) est un problème majeur de la détection, certains objets pouvant aussi être partiellement cachés ou flous. Les contours des objets correspondant le plus souvent à des discontinuités locales de la fonction des niveaux de gris, ils ont très souvent entraîné la nécessité de différencier un signal bruité. A noter que les approches "contours" et "régions" sont en fait duales, dans le sens où une région définit une ligne par son contour et qu'une ligne fermée détermine une région.

Les approches Gradient et Laplacien

Soit donc I(x,y) un signal bruité de dimension 2 représentant une image naturelle. Les contours correspondent aux discontinuités d'ordre 0 de I. Cette hypothèse peut cependant être prise en défaut par certains contours associés aux discontinuités d'ordre 1 de I, appelés également "contours en toit". On réalise classiquement l'extraction des contours par deux méthodes :

- Calcul du Laplacien et détermination des passages par zéro; c'est l'approche dérivée seconde.
- Calcul du Gradient et extraction des extrêmas locaux de la norme; c'est l'approche dérivée première.

Les nouveaux systèmes de reconnaissance des formes permettent à partir d'un bitmap saisi par scanner de détecter les formes contenues dans une image. C'est la première étape sur le chemin de la vision artificielle. Cet article se propose de présenter, parmi toutes les techniques de détection des contours dans les images, celles utilisées dans des systèmes de vision opérationnels.

Dans ce qui suit, nous appelerons Grad(I) le gradient :

```
Grad(I) = (dI(x,y)/dx,dI(x,y)/dy)t
```

et Lap(I) le laplacien :

```
Lap(I) = (d »I(x,y)/d «x)+(d »I(x,y)/d »y)
```

Dans le cas d'une approche dérivée première, le but du calcul est de trouver la direction selon laquelle la variation locale des niveaux de gris est maximale (direction du gradient), et l'intensité de cette variation (norme du gradient). L'extraction des contours s'effectue par sélection des points de norme de gradient élevés, grâce aux deux étapes suivantes :

• Calcul de la norme du gradient :
avec la distance euclidienne

```
Ng(x,y) = ((dI/dx)» + (dI/dy)»)^(1/2)
```

ou avec la distance de la valeur absolue

```
Ng(x,y) = abs(dI/dx)  + abs(dI/dy)
```

• Détermination des points de gradient élevés :
On calcule une image O(x,y) telle que,

```
O(x,y) = 1 si Ng(x,y) > s,
```

s étant un seuil fixé a priori

```
O(x,y) = 0 sinon
```

Malheureusement, il n'existe pas toujours de seuils permettant de détecter les points du contour sans obtenir aussi ceux dûs au bruit. Un moyen de tourner cette difficulté consiste à rechercher non pas les points de norme de gradient élevé mais les extrêmas locaux de la norme du gradient (passage par zéro de la dérivée seconde dans la direction du gradient), et d'éliminer les points de gradient trop faibles par un algorithme de seuillage.

Dans le cas d'une approche dérivée seconde on doit disposer de la valeur du laplacien en chaque point de l'image, et l'on considère que les points de contour correspondent au passage par zéro du laplacien. L'extraction de ces passages par zéro s'effectue en deux étapes :

• Détermination d'une image de polarité :
On calcule une image P(x,y) telle que,

```
P(x,y) = 0 si Lap(I) > 0
et P(x,y) = 1 sinon
```

• Détermination des points de passage par zéro :
On calcule une image O(x,y) telle que,

```
O(x,y) = 1 si (x,y) est un point de
transition 0-1 ou 1-0 dans P
O(x,y) = 0 sinon
```

Une dernière étape de seuillage s'avère bien souvent indispensable afin d'éliminer les points de faible gradient. Bien que basées sur des approches différentes, ces deux méthodes fournissent en pratique des résultats très proches.

Le filtrage linéaire

Il nous suffit donc maintenant de trouver des méthodes pour calculer Grad(I) et Lap(I) le plus simplement possible. Une bonne approche consiste à rechercher des filtres linéaires en permettant l'approximation.

Soit f un tel filtre, un théorème élémentaire de la théorie du signal nous apprend que la dérivée de l'image convoluée avec f est égale à l'image convoluée avec la dérivée de f. Grâce à cette propriété, le problème se ramène à la détermination d'un filtre de lissage de réponse Lis(x,y), et l'on peut calculer les dérivées premières (resp. seconde), en convoluant directement avec (dLis(x,y)/dx) et (dLis(x,y)/dy) .(resp. (d»Lis(x,y)/d»x) et (d»Lis(x,y)/d»y)). Nous proposons dans ce qui suit un ensemble de filtres correspondant à ces critères; ces filtres étant linéaires, leur implémentation donne lieu à des algorithmes de faible complexité. Un des premiers filtres de détection dû à Prewitt se présente comme suit :

```
Lis= | 1 1 |   et dLis/dx= | 1 0-1 |   dLis/dy= | -1-1-1 |
     | 1 1 |               | 1 0-1 |            |  0 0 0 |
                           | 1 0-1 |            |  1 1 1 |
```

Dans le cas d'une approche dérivée première, il peut être judicieux dans certains cas de calculer le gradient par une série de dérivations bidirectionnelles. C'est-à-dire de définir un masque de dérivation pour chacune des orientations possibles du contour, et de retenir en chaque point l'orientation selon laquelle le gradient est maximum. De tels masques ont été proposés par Kirsh, ils sont obtenus par la rotation d'un

filtre de dérivation selon 8 orientations (*figure 2*) :

```
N  | 5 5 -3 |   | 5 5 -3 |
   | -3 0 -3 | N-W | 5 0 -3 |
   | -3-3-3 |   | -3-3-3 |

W  | 5-3-3 |   | 5 5 -3 |
   | -3 0-3 | S-W | 5 0 -3 |
   | 5-3-3 |   | -3-3-3 |
```

Le coût élevé de l'implantation de ces filtres fait qu'ils sont assez peu utilisés. Ils peuvent cependant se révéler redoutablement efficaces dans des zones de l'image où plusieurs contours se rejoignent.

Les filtres que l'on dit séparables

Quand on décide d'approximer le gradient ou le laplacien de l'image par dérivation selon les deux directions, des problèmes de temps de calcul commencent à se poser. En effet pour une image (Xd x Yd), le coût d'une convolution de noyau (p x p) est (p» x Xd x Yd). Si l'on approxime les dérivées partielles par un filtre (p x p) il en coûte au total (2 x p» x Xd x Yd).

Ces remarques conduisent à l'introduction de filtres à réponses séparables selon les directions x et y. Le filtrage séparable comporte en effet bien des avantages et permet en particulier :

- la réduction du temps de calcul d'une convolution de p» à 2 x p.
- de prendre en compte des bruits de caractéristiques différentes selon les deux axes.

Deux autres atouts majeurs sont de pouvoir généraliser directement le filtre à une dimension quelconque, et la possibilité de filtrage récursif.

De plus, dans le cas où Lis(x,y) est un filtre séparable on a :

```
I(x,y) * Lis(x,y) = I(x,y) * (LisX(x)
X LisY(y))
```

Or, la convolution d'un signal par le produit de deux filtres séparables est égale à la convolution par le deuxième filtre du signal convolué par le premier filtre :

```
I(x,y) * Lis(x,y) = (I(x,y) * LisX(x)) *
LisY(y)
```

La cascade des filtres LisX et LisY permet donc d'implémenter Lis. La dérivation et le lissage d'un signal de dimension deux est ainsi réduit au cas monodimensionnel.

Le premier opérateur utilisant le filtrage séparable a été introduit par Sobel (*figure 3*) :

```
H1 | 1 0-1 |   | -1-2-1 |
   | 2 0-2 | et H2 = | 0 0 0 |
   | 1 0-1 |   | 1 2 1 |
```

A titre d'exemple on obtient avec cet opérateur le calcul suivant pour Ng, X1 à X8 représentant les valeurs des pixels au voisinage du point étudié X (*figure 4*) :

X1 X2 X3		
	X4 X X5	
		X6 X7 X8

```
Ng = (abs( X1 + 2X2 + X3 - X6 - 2X7 -X8) +
abs( X3 + 2X5 + X8 -X1 -2X3 -X6) )/8
```

On notera que Ng est normalisé pour correspondre au gradient par pixel. Pour ce qui est du laplacien, nous recommandons l'opérateur suivant :

```
Lap = (8X - X1 - X2 - X3 - X4 - X5 - X6
- X7 - X8)/3
```

En général, le calcul de la dérivée seconde est plus sensible au bruit, et une étape de seuillage des passages par zéro est indispensable. Cependant, la localisation des contours obtenus est pratiquement la même. Une approche de type Laplacien ayant tendance à arrondir les angles, on choisit de préférence une approche gradient si des angles aigus sont présents dans l'image.

Une nouvelle approche

Il est souvent difficile d'obtenir des traits de contours continus et bien nets par les approches énumérées plus haut. La méthode proposée ici reste fiable même dans le cas où la forme est particulièrement irrégulière ou dès que le bruit est important dans l'image. Elle permet assez simplement le tracé de contours continus et anti-aliasés.

Pour ce faire, on a recours à un opérateur de normalisation p, défini par :

```
p = (Gmin - Ng)/(Gmax - Gmin)
si (Gmax - Gmin > k)
p = (Gmin - Ng)/k sinon
```

Gmin et Gmax sont les valeurs maximales et minimales du gradient dans le voisinage du pixel, tandis que k est une constante qui permet de distinguer les discontinuités des changements continus. Sa valeur dépend des images et, pour notre part, nous avons obtenu de bons résultats avec k = 10.

Si nous appelons d la largeur du contour en pixel, l'image finale O(x,y) des contours est obtenue comme suit :

```
Pour x allant de 0 à  la largeur de
l'image
  faire
Pour y allant de 0 à  la hauteur de
l'image
     faire
X = I(x,y);
     O(x,y) = Cf + (Cc - Cf) *
DChange((X-p)/Ng);
  fin
     fin
```

DChange(t) étant la fonction de changement de densité :

```
DChange(t) = 1 si (0 < |t| <  d/2)
DChange(t) = 0 sinon
```

Cf et Cc sont respectivement la couleur du fond et la couleur du contour. L'application brutale de cette méthode conduit cependant à des effets d'aliasing (marches d'escaliers) qui peuvent s'avérer gênants dans certaines applications.

Pour remédier très simplement à ce petit inconvénient, on utilise dans DChange(t) une fonction linéaire (ou d'ordre plus élevé) à la place d'une fonction à deux niveaux.

Un dernier problème persiste cependant dans les régions de l'image où Ng est proche ou égal à zéro. Une solution consiste à rajouter un traitement qui donne à O(x,y) une valeur fixe (par exemple Cf) lorsque Ng est inférieur à une valeur seuil.

Nous présentons ici des résultats obtenus en appliquant la méthode précédemment décrite.

On obtient ainsi de bons résultats, au moins meilleurs que ceux des méthodes plus anciennes sur une grande variété d'images (*figures 5 et 6*).

Figure 5 : Image originale, avant le traitement complet par changement de densité.

Traitement en aval

La plupart des applications ne peuvent cependant pas se contenter d'une telle description de l'image car elle reste matricelle, et il faut en général la passer sous forme d'une liste de contours chaîne afin de générer une structure de données propre. G. Giraudon a mis au point un algorithme de chaînage particulièrement efficace, en reprenant une idée déjà utilisée sur des problèmes de restauration d'image : la décomposition d'un pavé (3x3) centré autour du pixel en passé, présent et futur.

Sa méthode se divise en trois étapes :

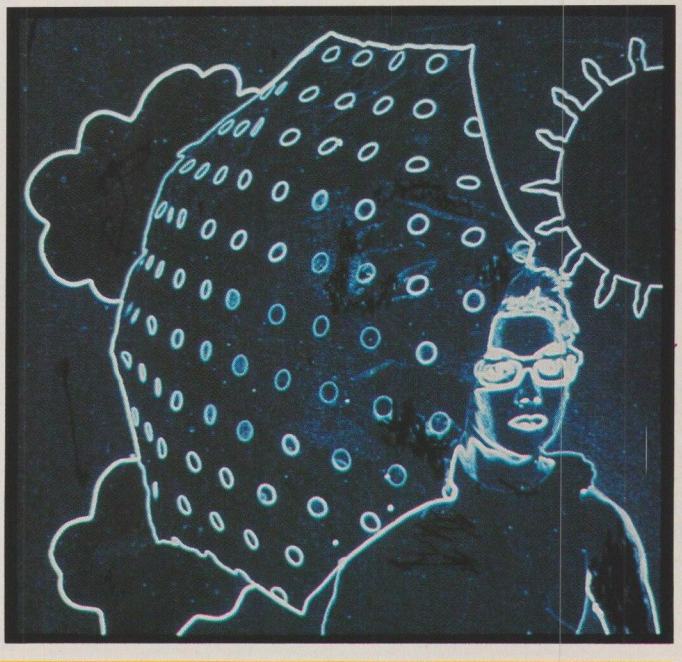
- Création des chaînes en parallèle, en une seule passe par balayage de l'image avec un pavé (3x3); les chaînes sont créées par un simple jeu de pointeurs.
- Fusion des chaînes, quand cela est possible, pour en diminuer le nombre.
- Elimination des chaînes restantes suivant divers critères (seuil de longueur de chaîne, par exemple).

Une méthode d'approximation polygonale en aval du chaînage des contours peut permettre d'obtenir des segments de droite.

Nous avons vu au cours de cet article comment passer de la représentation matricielle d'une image (un signal I(x,y)) à une description comme un assemblage structuré de primitives élémentaires (segments de droite). Mais cette étape n'est qu'un problème particulier de la vision artificielle dont l'étude générale sort du contexte de cet article.

La réalisation d'un logiciel de reconnaissance complet pouvant demander plusieurs mois/ingénieur, nous proposons à titre indicatif un exemple d'implémentation d'un mo-

Figure 6 : Image résultante, correspondant aux contours de l'écran de départ.



LA DÉTECTION DES CONTOURS DANS LES IMAGES

dule de détection de contours volontairement limité (par manque de place) aux images binaires.

Il s'agit donc d'un listing relativement court et simplifié, le contour étant dans ce cas réduit à l'ensemble des points de l'objet dont le voisinage contient des points appartenant à l'extérieur de l'objet. Il devrait néanmoins constituer une bonne base pour permettre l'expérimentation de nouveaux opérateurs. □

BIBLIOGRAPHIE

- A. Rosenfeld and A Kak. Digital image processing . Academic Press: New York
- D. Marr and E. Hildreth. Theory of edge detection . Proc. Royal Soc. Lond.
- J. Shen . An optimal linear operator for edge detection . Conference on vision and pattern recognition, USA, June 1986 . IEEE
- R. Kohler . A segmentation system based on thresholding . Computer Graphics and Image Processing, 15:318-338, 1981 .
- G. Giraudon . An efficient edge chaining algorithm . 5th Scandinavian conference on image analysis, June 1987 .
- M. Berthod . Un nouvel algorithme d'approximation polygonale . Technical Report, INRIA, 1987 .


```
/* algorithme de détection de contour
dans des images binarisées */


#include <values.h>
#include "bitmap.h"
#include "global.h"
#include "pxl-outline.h"
#include "report.h"

#undef  STANDALONE

typedef enum
{
left = 0, top = 1, right = 2, bottom = 3,
no_edge = 4
} edge_type;

#define NUM_EDGES  4
#define START_EDGE  top

static pixel_outline_type
find_one_outline (gf_char_type,
edge_type, unsigned row, unsigned col,
bitmap_type * marked);

static void next_outline_edge
(bitmap_type, edge_type *, unsigned *row,
unsigned *col,
coordinate_type * corner_pixel);
```

```
static edge_type
next_unmarked_outline_edge (unsigned row,
unsigned col, edge_type edge, bitmap_type
character, bitmap_type marked);

static boolean is_outline_edge (edge_type
edge, bitmap_type character, unsigned
row, unsigned col);

static void mark_edge (edge_type edge,
unsigned row, unsigned col, bitmap_type *
marked);

static boolean is_marked_edge (edge_type
edge, unsigned row, unsigned col,
bitmap_type marked);

static edge_type next_edge (edge_type
edge);

static string edge_name (edge_type edge);


/* routines principales */

static void append_pixel_outline
(pixel_outline_list_type *,
pixel_outline_type);
static pixel_outline_type
new_pixel_outline ();
static void append_coordinate
(pixel_outline_type *, boolean is_corner,
int x, int y);


/* routines d'écriture.  */

static void log_outline_list
(pixel_outline_list_type);

static void print_outline_list (FILE *,
pixel_outline_list_type);

static bounding_box_type get_outline_bb
(pixel_outline_list_type);

static coordinate_type
cartesian_to_bitmap_coordinate (

coordinate_type cartesian,

bounding_box_type cartesian_bb);

static void print_bytemap (FILE *,
bitmap_type);
static void print_bounded_bytemap (FILE*,
bitmap_type, bounding_box_type);



/* On parcourt le bitmap de haut en bas
et de gauche à droite à la recherche d'un
pixel non marqué. Les pixels du contour
```

```
ainsi trouvés sont placés dans une liste
et sont retournés en fin de fonction  */

pixel_outline_list_type
find_outline_pixels (gf_char_type
gf_char)
{
  pixel_outline_list_type outline_list;
  bitmap_type bitmap = get_bitmap
(gf_char);
  bitmap_type marked = new_bitmap
(bitmap_dimensions (bitmap));
  unsigned this_row, this_col;

  outline_list.length = 0;
  outline_list.data = NULL;

  for (this_row = 0; this_row <
bitmap_height (bitmap); this_row++)
    {
      for (this_col = 0; this_col <
bitmap_width (bitmap); this_col++)
  {
    edge_type edge =
next_unmarked_outline_edge (this_row,
this_col,

  START_EDGE, bitmap, marked);
    if (edge != no_edge)
      {
        append_pixel_outline
(&outline_list,
        find_one_outline (gf_char, edge,
this_row, this_col, &marked));
      }
  }
    }
  log_outline_list (outline_list);
  log1 ("%u outline(s) in all.\n",
outline_list.length);

  return outline_list;
}


static pixel_outline_type
find_one_outline (gf_char_type gf_char,
edge_type starting_edge, unsigned
original_row, unsigned original_col,
bitmap_type * marked)
{
  bitmap_type bitmap = get_bitmap
(gf_char);
  pixel_outline_type outline =
new_pixel_outline ();
  unsigned previous_row = original_row+1;
  unsigned previous_col = original_col+1;
  unsigned row = original_row;
  unsigned col = original_col;

  edge_type edge = starting_edge;

  do
    {
      coordinate_type corner_pixel;

            if (!(row == previous_row &&
col == previous_col))
        append_coordinate (&outline,
false,col, get_char_max_row (gf_char) -
row);
      previous_row = row;
      previous_col = col;
      mark_edge (edge, row, col, marked);
      next_outline_edge (bitmap, &edge,
&row, &col, &corner_pixel);

      if (corner_pixel.x >= 0 &&
corner_pixel.y >= 0)
        append_coordinate (&outline,
true, corner_pixel.x,
                    get_char_max_row
(gf_char) - corner_pixel.y);
    }
  while (!(row == original_row && col ==
original_col));

  return outline;
}

/* On se déplace à 'l'exterieur' d'un
contour dans le sens horaire,et à
`l'intérieur' dans le sens inverse.  */

static void
next_outline_edge (bitmap_type character,
edge_type * edge,unsigned *row, unsigned
*col,coordinate_type * corner_pixel)
{
  unsigned original_row = *row;
  unsigned original_col = *col;

*corner_pixel=((coordinate_type){-1,-1});

  switch (*edge)
    {
    case left:
      if (*row != 0)
 {
   if (*col != 0)
     {
       if (is_outline_edge (bottom,
character, *row - 1, *col - 1))
         {
           *corner_pixel =
((coordinate_type)
                        {
             *col, *row - 1
           });
           (*row)--;
           (*col)--;
           *edge = bottom;

           break;
         }
     }
   if (is_outline_edge (left, character,
*row - 1, *col))
     {
       (*row)--;
     }
 }
      break;

    case top:


      if (*col != bitmap_width
(character) - 1)
 {
   if (*row != 0)
     {
```

```
        if (is_outline_edge (left,
character, *row - 1, *col + 1))
          {
            *corner_pixel =
((coordinate_type)
                              {
              *col + 1, *row
            });
            (*row)--;
            (*col)++;
            *edge = left;
            break;
          }
      }
    if (is_outline_edge (top, character,
*row, *col + 1))
      {
        (*col)++;
      }
  }
      break;

    case right:
      if (*row != bitmap_height
(character) - 1)
  {
    if (*col!=bitmap_width (character)-1)
      {
        if (is_outline_edge (top,
character, *row + 1, *col + 1))
          {
            *corner_pixel =
((coordinate_type)
                              {
              *col, *row + 1
            });
            (*row)++;
            (*col)++;
            *edge = top;
            break;
          }
      }
    if (is_outline_edge (right,
character, *row + 1, *col))
      {
        (*row)++;
      }
  }
      break;

    case bottom:
      if (*col != 0)
  {
   if (*row !=bitmap_height(character)-1)
      {
        if (is_outline_edge (right,
character, *row + 1, *col - 1))
          {
            *corner_pixel =
((coordinate_type)
                              {
              *col - 1, *row
            });
            (*row)++;
            (*col)--;
            *edge = right;
            break;
          }
      }
    if (is_outline_edge (bottom,
character, *row, *col - 1))
      {
        (*col)--;
      }
  }
      break;

    case no_edge:
      fatal ("Found a no_edge in
next_outline_edge");

    default:
      fatal ("Bad edge value %d in
next_outline_edge", *edge);

    }

/* Si l'on n'a pas trouvé un contour
adjacent ou un autre pixel alors on
cherche le contour suivant sur le pixel
courant.
*/
  if (*row == original_row && *col ==
original_col)
    {
      *edge = next_edge (*edge);
    }
}



static edge_type
next_unmarked_outline_edge (unsigned row,
unsigned col,edge_type starting_edge,
bitmap_type character, bitmap_type
marked)
{
  edge_type edge = starting_edge;

  assert (edge != no_edge);

  while (is_marked_edge (edge, row, col,
marked)
   || !is_outline_edge (edge, character,
row, col))
    {
      edge = next_edge (edge);
      if (edge == starting_edge)
        return no_edge;
    }

  return edge;
}


/* On verifie que le côté EDGE du pixel
de la position ROW, COL est un contour;
c'est à dire un pixel noir qui partage ce
coté avec un pixel blanc.  ROW et COL
doivent être a l'intérieur du bitmap
 */

static boolean
is_outline_edge (edge_type edge,
bitmap_type character,
         unsigned row, unsigned col)
{
  if (row < 0 || row >= bitmap_height
(character)
      || col < 0 || col >= bitmap_width
```

```
(character))
    fatal2 ("Bad bitmap position: row =
%u, col = %u", row, col);

/* si le pixel n'est pas noir, il ne fait
pas parti du contour  */
  if (bitmap_pixel (character, row, col)

== WHITE)
    return false;

  switch (edge)
    {
    case left:
      return col == 0 || bitmap_pixel
(character, row, col - 1) == WHITE;

    case top:
      return row == 0 || bitmap_pixel
(character, row - 1, col) == WHITE;

    case right:
      return col == bitmap_width
(character) - 1
  || bitmap_pixel (character, row, col+1)
== WHITE;

    case bottom:
      return row == bitmap_height
(character) - 1
  || bitmap_pixel (character, row + 1,
col) == WHITE;

    case no_edge:

    default:
      fatal ("Bad edge value of %d in
is_outline_edge", edge);
    }
}


/* On marque EDGE si cela n'est pas déjà
fait, autrement c'est une erreur fatale .
*/

static void
mark_edge (edge_type edge, unsigned row,
unsigned col,
     bitmap_type * marked)
{
  if (is_marked_edge (edge, row, col,
*marked))
      fatal ("Already marked edge %s at
row %d col %d",edge_name (edge), row,
col);

  if (edge != no_edge)
{
bitmap_pixel (*marked, row, col)|=1 <<
edge;
    }
}


static boolean
is_marked_edge (edge_type edge, unsigned
row, unsigned col, bitmap_type marked)
{
  return edge == no_edge
         ? false
         : bitmap_pixel (marked, row,
col) & (1 << edge);
}


static edge_type
next_edge (edge_type edge)
{
  return edge == no_edge ? edge : (edge +
1) % NUM_EDGES;
}


static string
edge_name (edge_type edge)
{
  static string edge_names[] =
  {"left", "top", "right", "bottom",
"no_edge"};

  return edge_names[edge];
}

static void
append_pixel_outline
(pixel_outline_list_type * outline_list,
                pixel_outline_type outline)
{
  outline_list->length++;
  outline_list->data = reallocate
(outline_list->data,
                outline_list->length *
sizeof (pixel_outline_type));
outline_list->data[outline_list->length-
1] = outline;

  log1 ("Length = %u.\n",
outline.length);
}


/* Ecrit OUTLINE_LIST  dans' log file' .
On suppose que le fichier est déjà
ouvert*/

static void
log_outline_list (pixel_outline_list_type
outline_list)
{
  if (logging)
    print_outline_list (log_file,
outline_list);
}

/* Ecrit OUTLINE_LIST  dans 'FILE' comme
bitmap. On suppose que le fichier est
déjà ouvert  */

static void
print_outline_list (FILE *file,
pixel_outline_list_type outline_list)
{
  unsigned this_outline_list,
this_outline;
  bounding_box_type outline_bb =
get_outline_bb (outline_list);
  bitmap_type bytemap = new_bitmap
(bb_to_dimensions (outline_bb));
  char pixel_char = 'a';
```

LA DÉTECTION DES CONTOURS DANS LES IMAGES

```
  for (this_outline_list = 0;
this_outline_list < outline_list.length;
       this_outline_list++)
    {
      pixel_outline_type outline =
outline_list.data[this_outline_list];

      for (this_outline = 0; this_outline
< outline.length; this_outline++)
  {
    coordinate_type here =
cartesian_to_bitmap_coordinate

(O_COORDINATE (outline, this_outline),

outline_bb);

    char bytemap_char = get_bitmap_pixel
(bytemap, here);

    if (bytemap_char == -1)
      {
        fatal2 ("Bad bytemap position:
(%d,%d)\n", here.x, here.y);
      }
    set_bitmap_pixel (bytemap, here,
                      bytemap_char == WHITE
                      ? pixel_char
                      : pixel_char -
bytemap_char + '0');
  }
      switch (pixel_char)
  {
  case 'z':
    pixel_char = 'A';
    break;
  case 'Z':
    fatal ("Ran out of letters for
logging outline");
  default:
    pixel_char++;
  }
    }
  print_bytemap (file, bytemap);
  fprintf (file, "\n");
}


/* On parcourt tous les pixels de
OUTLINE_LIST pour trouver les coordonnées
de la boite englobante */

static bounding_box_type
get_outline_bb (pixel_outline_list_type
outline_list)
{
  unsigned this_outline_list,
this_outline;
  bounding_box_type answer =
  {MAXINT, -MAXINT, MAXINT, -MAXINT};

  for (this_outline_list = 0;
this_outline_list < outline_list.length;
       this_outline_list++)
    {
      pixel_outline_type outline =
outline_list.data[this_outline_list];

      for (this_outline = 0; this_outline
< outline.length; this_outline++)
  {
```

```
    coordinate_type coordinate =
O_COORDINATE (outline, this_outline);

    if (coordinate.y < get_min_row
(answer))
            get_min_row (answer) =
coordinate.y;

    if (coordinate.y > get_max_row
(answer))
            get_max_row (answer) =
coordinate.y;

    if (coordinate.x < get_min_col
(answer))
            get_min_col (answer) =
coordinate.x;

    if (coordinate.x > get_max_col
(answer))
            get_max_col (answer) =
coordinate.x;
  }
    }
  return answer;
}


static coordinate_type
cartesian_to_bitmap_coordinate
(coordinate_type cartesian_coordinate,
                        bounding_box_type
cartesian_bb)
{
  coordinate_type answer =
cartesian_coordinate;

  answer.y = get_max_row (cartesian_bb) -
cartesian_coordinate.y;

  return answer;
}


static void
print_bytemap (FILE * f, bitmap_type
bytemap)
{
  print_bounded_bytemap (f, bytemap,
get_bitmap_bounding_box (bytemap));
}

static void


print_bounded_bytemap (FILE * file,
bitmap_type bytemap, bounding_box_type
bb)
{
  unsigned this_row, this_col;

  fprintf (file, "Printing bytemap
between (%u,%u) and (%u,%u):\n",
     get_min_col (bb), get_min_row (bb),
     get_max_col (bb), get_max_row (bb));
  for (this_row = get_min_row (bb);
this_row <= get_max_row (bb);
       this_row++)
    {
```

```
      for (this_col = get_min_col (bb);
this_col <= get_max_col (bb);
     this_col++)
  {
    char this_pixel = bitmap_pixel
(bytemap, this_row, this_col);
    putc (this_pixel != WHITE ?
this_pixel : ' ', file);
  }
      fprintf (file, "%u\n", this_row);
    }
}

pixel_outline_type
new_pixel_outline ()
{
  pixel_outline_type pixel_outline;

  pixel_outline.length = 0;
  pixel_outline.data = NULL;

  return pixel_outline;
}


void
append_coordinate (pixel_outline_type *
o, boolean is_corner, int x, int y)
{
  o->length++;
  o->data = reallocate (o->data, o-
>length * sizeof (outline_pixel_type));
  O_PIXEL_X (o->data[o->length - 1]) = x;
  O_PIXEL_Y (o->data[o->length - 1]) = y;
  O_PIXEL_CORNER (o->data[o->length - 1])
= is_corner;
}

#ifdef STANDALONE

FILE *log_file;
boolean logging = false;
boolean verbose = false;

static string filename_prefix =
"no_name";

static void parse_commandline (int argc,
string argv[]);


void
main (int argc, string argv[])
{
  string gf_filename;
  pixel_outline_list_type outline_list;
  unsigned this_char;

  parse_commandline (argc, argv);
  gf_filename =
concat(filename_prefix,".ima");
  if (!open_input_file (gf_filename))

    {
      fatal_perror (gf_filename);
    }
  log_file = checked_fopen (concat
(filename_prefix, ".log"), "w");

      gf_char_type gf_char =
*get_given_gf_char (this_char);
```

```
      report ("En cours de traitement :
%s\n", gf_filename);

            outline_list =
find_outline_pixels (gf_char);

          print_outline_list (stderr,
outline_list);


  exit (0);
}

#define OPTION_HELP "Options:
-ima <filename_prefix>: Give the prefix
for the image file (.ima).
-logging: If you want output details put
in a log file.
-verbose: Use if you want some extra
runtime information printed on terminal.
"

#define get_current_string_argument() \
   string_copy(argv[this_arg])

static void
parse_commandline (int argc, string
argv[])
{
 unsigned this_arg;

 if (argc <= 1)
   {
     printf ("Usage: %s <options>.\n",
argv[0]);
     printf (OPTION_HELP);
     exit (1);
   }

 for (this_arg = 1; this_arg < argc;
this_arg++)
   {
     if (*argv[this_arg] == '-')
       {
  argv[this_arg]++;

  else if (argument_is ("ima"))
          filename_prefix =
get_string_argument ();

  else if (argument_is ("logging"))
          logging = true;

  else if (argument_is ("verbose"))
          verbose = true;

  else
          fatal1 ("argument inconnu
`%s'", argv[this_arg]);
       }
   }
 if (filename_prefix == "no_name")
   fatal ("You didn't give us a image to
work with: use -ima <filename>");
}

#endif /* STANDALONE*/
```

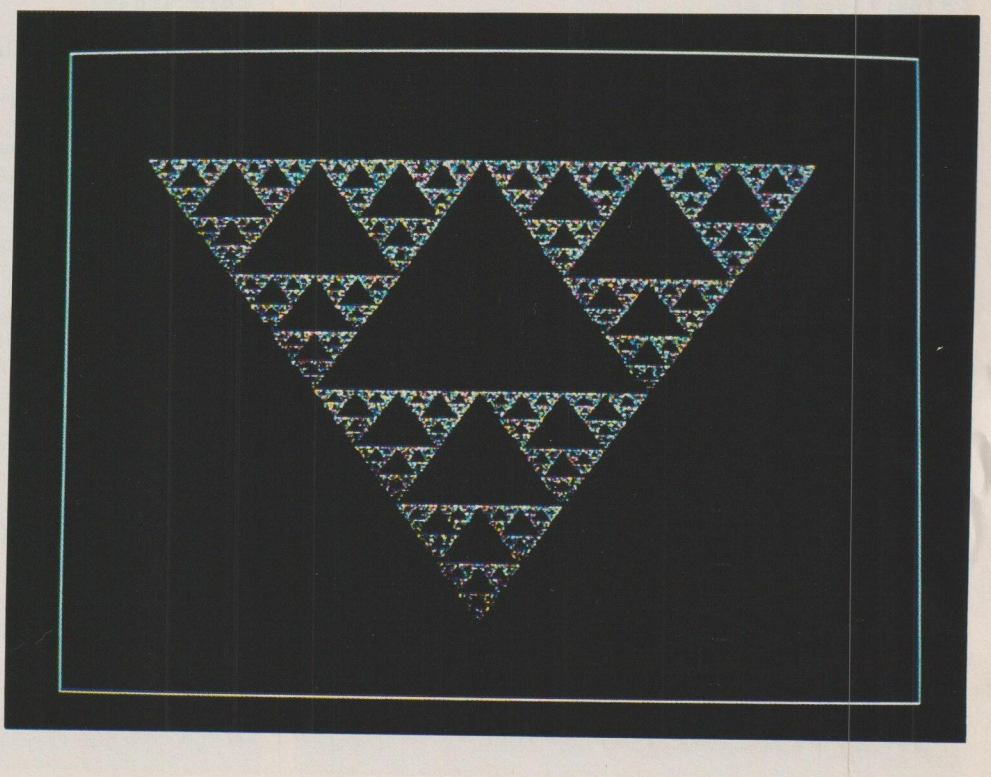
LA GÉOMÉTRIE FRACTALE

HENRY LEMBERG

Qui n'a jamais entendu parler de fractales ? La théorie, découverte et développée par le mathématicien Benoit Mandelbrot, popularisée par de superbes photos de courbes se répétant à l'infini dans un chaos qui semble ordonné, est, pour beaucoup, l'unique achèvement connu des mathématiques du XXème siècle. Pourtant, en dehors de la beauté visuelle d'une équation de fractale, on a peu développé jusqu'à présent de réelles applications qui permettraient de passer du simple jeu mathématique à l'outil technologique synonyme du véritable succès populaire.

Telle n'est pas l'optique de Michael Barnsley qui, par ses travaux commencés au milieu des années 80, a non seulement réussi à simplifier et à concrétiser ces notions de chaos dynamique, mais également à leur trouver des applications. Il est probable qu'elles soient utilisées dans un futur proche quand il s'agira de stocker un maximum d'images sur un vidéodisque en vue d'applications multimédias. L'idée de Michael Barnsley est, comme la plupart des idées géniales, simple au départ : lorsqu'on regarde un paysage, un champ de blé, une forêt, un arbre, une feuille, on est frappé par la ressemblance, à des échelles différentes, des éléments qui composent ces figures. A partir de deux ou trois traits, et si l'on sait les réassembler, il est possible de recomposer une branche de sapin, puis le sapin, la forêt toute entière, et, en codant ainsi plusieurs détails, on pourra recomposer un paysage dans son ensemble. Deux difficultés se présentent alors. La première est d'ordre culturel : comment reconnaître dans une figure ces petits éléments qui composent la figure entière ? Habitués depuis toujours à penser la géométrie comme un assemblage de lignes polygonales, peut-on facilement modifier sa perception et regarder une image comme un ensemble de fractales ?

Certes ce n'est pas immédiat, mais il est toujours permis d'essayer de changer ses habitudes, de la même façon que l'on passe d'un langage de programmation à un autre, où les déclarations, par exemple, s'expriment différemment. Il existe cependant des techniques qui, à partir d'une image digitalisée, permettent de trouver cette décomposition en figures élémentaires : séparation de couleurs, analyse spectrale ou de variation de texture, techniques qui se retrouvent dans le coloriage de films tournés en noir et blanc. Ces techniques sont à réserver à ceux qui voudront utiliser les idées de



Le triangle de Sierpinski. L'image, un ensemble de fractales, est décomposée en petites figures élémentaires.

Contrairement à l'algorithmique classique qui consiste à découper une figure en lignes droites régulières puis à lisser les polygones ainsi obtenus, il existe une autre géométrie, originale, inspirée de la théorie des fractales et qui trouve des applications concrètes quand il s'agit d'atteindre des taux de compression élevés pour le stockage d'images.

Barnsley pour la compression d'images. Les exemples que nous donnerons par la suite font un peu "cas d'école", mais ils sont toutefois représentatifs des énormes possibilités de cette géométrie. La seconde difficulté est d'ordre pratique : une fois l'image décomposée en petites figures élémentaires, comment la recomposer, et surtout, est-on assuré que l'on retrouvera en fin de compte une image identique à celle de laquelle on est parti ? C'est là que les idées simples (mêmes géniales) ne suffisent plus, et qu'un travail théorique est nécessaire pour s'assurer de la validité des algorithmes proposés. Le travail de Michael Barnsley consiste à coder des ensembles de nombres, appelés Iterated Function System (IFS), qui reproduisent les différentes fractales. Le théorème du collage nous assure de toujours trouver un ensemble d'IFS pour recomposer notre image et donne un algorithme pour le faire.

Le codage

La théorie IFS repose sur les transformations affines du plan (se reporter à l'encadré ci-contre pour les explications mathématiques), qui sont toutes de la forme T(x,y)= (ax+by+e, cx+dy+f), où (a,b,c,d,e,f) sont six nombres réels. Par exemple, une transformation de ce type fera passer le point de coordonnées (x,y) au point T(x,y)= (0.5x+0.33y-0.25, 0.33x-0.25y+0.5). De plus, seules les transformations affines contractives sont ici permises, c'est-à-dire une application réduisant la distance entre deux points. A partir d'une figure géométrique simple donnée, il est relativement aisé de trouver la transformation contractive qui permettra de passer à une figure identique, mais réduite. Ceci se réalise le plus simplement du monde à l'aide d'un solveur capable de résoudre des systèmes d'équations linéaires.

Par définition, un IFS est un ensemble de plusieurs transformations (T1,T2,...Tn), chaque transformation étant associée à une probabilité Pi (la somme de ces probabilités faisant bien entendu 1). L'encadré sur le choix des probabilités donne quelques exemples d'IFS pour des figures particulières. Le choix de ces probabilités n'est pas très important en ce qui concerne l'image (des théorèmes mathématiques viennent prouver qu'on récupérera toujours une image identique), mais il est crucial pour la rapidité avec laquelle on récupérera cette image. Si F est une image, et si Ti est un des composants de l'IFS (T1,T2,...Tn), un bon choix consiste à

TRANSFORMATION AFFINE DU PLAN : ÉQUATIONS MATHÉMATIQUES

Une transformation affine du plan peut se coder sous forme matricielle de la façon suivante:

$$T\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}$$

les coefficients (a, b, c, d, e, f) étant des nombres réels.
Le résultat se lit:

$$T(x, y) = (ax + by + e, cx + dy + f)$$

Géométriquement, une telle transformation consiste à effectuer une rotation et une homothétie (agrandissement ou retrécissement suivant chaque direction) de la figure (qui correspond à la partie $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$), puis à translater la figure dans le plan (partie $\begin{pmatrix} e \\ f \end{pmatrix}$).
Aussi, si l'on connait à l'avance l'angle de la rotation ϕ, et les coefficients d'homothétie α sur l'axe $\overrightarrow{Ox}$ et β sur l'axe $\overrightarrow{Oy}$, la transformation s'écrit

$$T\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \alpha\cos\phi & -\beta\sin\phi \\ \alpha\sin\phi & \beta\cos\phi \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}$$

Le problème inverse est aussi facile à traiter: connaissant la figure de départ et celle d'arrivée, qui a la même structure, mais qui est à une échelle réduite, comment trouver la matrice et le vecteur de translation qui permettent de passer de la première à la seconde? Il s'agit simplement de trouver les 6 inconnues (a, b, c, d, e, f) qui forment l'équation matricielle. Pour cela, il suffit de choisir 3 points de la première figure $(x_1, y_1), (x_2, y_2), (x_3, y_3)$, et 3 points de la seconde $(x'_1, y'_1), (x'_2, y'_2), (x'_3, y'_3)$ qui vont leur correspondre.
Trouver les 6 inconnues revient alors à résoudre le système d'équations

$$\begin{cases} ax_1 + by_1 + e = x'_1 \\ ax_2 + by_2 + e = x'_2 \\ ax_3 + by_3 + e = x'_3 \\ cx_1 + dy_1 + f = y'_1 \\ cx_2 + dy_2 + f = y'_2 \\ cx_3 + dy_3 + f = y'_3 \end{cases}$$

CHOIX DES PROBABILITÉS ASSOCIÉES À UN SYSTÈME IFS

Un choix des probabilités associées à un système IFS $(T_1, T_2, \ldots, T_n)$ qui travaille sur un figure F consiste à prendre

$$\frac{\mathcal{A}(T_i(F))}{\mathcal{A}(F)}$$

où $\mathcal{A}(F)$ désigne l'aire de la figure . (Cela représente la proportion de points aléatoires qui se retouveront dans la partie $T_i(F)$ de la figure).
On peut monter que si T_i est teprésenté par la matrice $\begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix}$ alors

$$\mathcal{A}(T_i(F)) = |\det T_i| = |a_i d_i - c_i b_i|$$

Aussi, la probabilité p_i à choisir sera

$$p_i = \frac{|a_i d_i - c_i b_i|}{\sum_{k=1}^{n} |a_k d_k - c_k b_k|}$$

Si jamais un des nombres p_i est nul, il faudra le remplacer par une petite valeur (0,001 par exemple) et ajuster les autres probabilités de façon à ce que leur somme vaille 1.

Image fractale d'une fougère.
Une application de cette géométrie : le stockage d'images.

prendre le quotient aire(Ti(F))/aire(F). Le théorème du collage montrera que ces surfaces ne dépendent en fait que des 6 nombres qui définissent chacune des transformations Ti.

Le jeu du chaos

Nous voilà prêt à donner l'algorithme qui permettra de retrouver, à partir du codage de l'IFS (T1,P1;T2,P2;...Tn,Pn), une figure donnée. Celui-ci est élémentaire :

1. Choisir N (nombre d'itérations) Initialiser x=0,y=0
2. Choisir un des nombres i de {1,2,...n} avec la probabilité Pi.
3. Calculer (x',y')= Ti(x,y)
4. Poser x:=x', y:=y'
5. Tracer le point (x,y)
6. Revenir à 2, N fois.

Le fait que cet algorithme, simplissime mais aléatoire, permette de toujours retrouver sinon exactement la même figure, tout au moins une figure quasiment identique, est assuré par les travaux du mathématicien John Elton en théorie ergodique. La principale difficulté consiste donc à trouver les transformations affines correctes qui composent un IFS.

Le théorème du collage

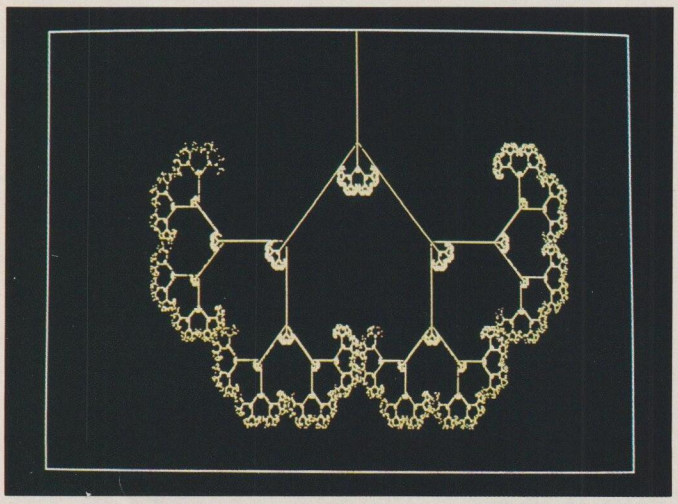
Le théorème du collage est une méthode permettant de trouver la ou les transformations affines contractives qui représentent le codage IFS. Sans donner sa formulation mathématique précise, le sens global de son énoncé est le suivant : pour qu'un système d'IFS (T1,T2,...,Tn) donne, en utilisant le jeu du chaos, une image semblable à une figure F donnée au départ, il faut trouver les transformations sur F telles que

l'union des Ti(F) recouvre quasiment F. Ce théorème ne fournit aucune technique pour y arriver, et il faudra utiliser un solveur pour résoudre les systèmes d'équations qui définissent chacune des transformations affines. Pourtant, sa signification est importante; même si le codage n'est qu'approximatif (avec toutefois une bonne approximation...), on récupérera toujours une copie satisfaisante de l'image originale. Plus encore, de légères perturbations du codage ne changent en rien la qualité du rendu obtenu. Ceci n'est pas sans rappeler les mémoires associatives des réseaux neuronaux, elles aussi insensibles aux perturbations.

Ces techniques d'encodage ont des applications pratiques car elle permettent des taux de compression d'images importants. Par exemple, la photo représentant une jeune fille bolivienne qui fait la couverture de l'ouvrage de référence a été codée sur 2 Kb seulement en utilisant un IFS de 120 transformations. D'autres images, plus fractales, comme un ensemble de nuages ou un paysage côtier, ont été codées sur 500 et 100 bits respectivement. L'ouvrage dont s'inspire cet article est une excellente introduction à la théorie du chaos dynamique et fait enfin comprendre, grâce à la technique IFS, comment réaliser des fractales. Il va plus loin encore en donnant les démonstrations mathématiques des théorèmes qui sont cités dans le présent article, et développe les notions de dimensions fractales ou les ensembles de Julia ou Mandelbrot. Il faudra toutefois avoir le niveau d'un DEUG en mathématiques ou d'une classe préparatoire pour en profiter.

Commentaires sur le programme

Le programme proposé met en oeuvre l'algorithme décrit plus haut. Il est écrit en Turbo Pascal 6, mais n'utilise aucune structure d'objet et peut être facilement réécrit en n'importe



L'arbre fractal.
Pour recomposer la figure, il faut trouver un ensemble d'IFS.

PROGRAMME FRACTAL

```
1. program fractal;
2. Uses Graph,Crt,Foug;

3. var x,y,nouvelX,nouvelY:real;
       n:integer;
       k,nbiter:longint;


4. procedure InitGraphisme;
   var GrPilote,GrMode,CodeErreur:integer;

 Begin
 GrPilote:=Detect;
 InitGraph(GrPilote,GrMode,'d:\tp\bgi');
 CodeErreur:=GraphResult;
 If CodeErreur=GrOk Then
         Rectangle(0,0,GetMaxX,GetMaxY)
    else writeln('Erreur');
 end;



5. function min(p:proba;x:real):integer;
   var k:integer;
       resul,y:real;
       fini:boolean;

  begin
  fini:=false;
  k:=1;
  resul:=abs(x-p[k]);
  while not(fini) do begin
       y:=abs(x-p[k+1]);
       if y<resul then begin
               resul:=y;
               k:=k+1;
              if k=Ntrans then fini:=true
                     end
              else fini:=true
              end;
  min:=k
  end;



   Begin
6.  InitGraphisme;
7.  Randomize;
8.  nbiter:=100000;
9.  x:=0;y:=0;

10. for k:=1 to nbiter do
    begin
11. n:=min(p,random);
12. nouvelX:=T[n][1,1]*x+T[n][1,2]*y+Tr[n][1];
      nouvelY:=T[n][2,1]*x+T[n][2,2]*y+Tr[n][2];
13. x:=nouvelX;
      y:=nouvelY;

15. PutPixel(trunc(x*Xechelle+Xoffset),
                    trunc(y*Yechelle+Yoffset),14)
16. end;

17. readln;
18. CloseGraph
19. end.
```

LA GÉOMÉTRIE FRACTALE

PROGRAMME DU TRIANGLE

```
UNIT Sierp;

INTERFACE
Const Ntrans=3;

Const Xechelle=250;
      Yechelle=250;
      Xoffset=80;
      Yoffset=60;


Type mat=array[1..2,1..2] of real;
     vect=array[1..2] of real;
     proba=array[1..Ntrans] of real;
Var T:array[1..Ntrans] of mat;
    Tr:array[1..Ntrans] of vect;
    p:proba;

IMPLEMENTATION
Begin
T[1][1,1]:=0.5;
T[1][1,2]:=0.0;
T[1][2,1]:=0.0;
T[1][2,2]:=0.5;

T[2][1,1]:=0.5;
T[2][1,2]:=0.0;
T[2][2,1]:=0.0;
T[2][2,2]:=0.5;

T[3][1,1]:=0.5;
T[3][1,2]:=0.0;
T[3][2,1]:=0.0;
T[3][2,2]:=0.5;

Tr[1][1]:=0; Tr[1][2]:=0;
Tr[2][1]:=1; Tr[2][2]:=0;
Tr[3][1]:=0.5;Tr[3][2]:=0.5;

p[1]:=0.34;p[2]:=0.67;p[3]:=1.0;

end.
```

PROGRAMME DE L'ARBRE

```
UNIT Arbre;

INTERFACE
Const Ntrans=4;

Const Xechelle=950;
      Yechelle=950;
      Xoffset=300;
      Yoffset=-100;


Type mat=array[1..2,1..2] of real;
     vect=array[1..2] of real;
     proba=array[1..Ntrans] of real;
Var T:array[1..Ntrans] of mat;
    Tr:array[1..Ntrans] of vect;
    p:proba;

IMPLEMENTATION
Begin
T[1][1,1]:=0.0;
T[1][1,2]:=0.0;
T[1][2,1]:=0.0;
T[1][2,2]:=0.5;

T[2][1,1]:=0.10;
T[2][1,2]:=0;
T[2][2,1]:=0;
T[2][2,2]:=0.10;

T[3][1,1]:=0.42;
T[3][1,2]:=-0.42;
T[3][2,1]:=0.42;
T[3][2,2]:=0.42;

T[4][1,1]:=0.42;
T[4][1,2]:=0.42;
T[4][2,1]:=-0.42;
T[4][2,2]:=0.42;


Tr[1][1]:=0; Tr[1][2]:=0;
Tr[2][1]:=0; Tr[2][2]:=0.2;
Tr[3][1]:=0; Tr[3][2]:=0.2;
Tr[4][1]:=0; Tr[4][2]:=0.2;

p[1]:=0.05;p[2]:=0.20;p[3]:=0.60;p[4]:=1;

end.
```

PROGRAMME DE LA FOUGÈRE

```
UNIT Foug;

INTERFACE
Const Ntrans=4;

Const Xechelle=640;
      Yechelle=250;
      Xoffset=0;
      Yoffset=-125;


Type mat=array[1..2,1..2] of real;
     vect=array[1..2] of real;
     proba=array[1..Ntrans] of real;
Var T:array[1..Ntrans] of mat;
    Tr:array[1..Ntrans] of vect;
    p:proba;

IMPLEMENTATION
Begin
T[1][1,1]:=0.0;
T[1][1,2]:=0.0;
T[1][2,1]:=0.0;
T[1][2,2]:=0.16;

T[2][1,1]:=0.20;
T[2][1,2]:=-0.26;
T[2][2,1]:=0.23;
T[2][2,2]:=0.22;
```

```
T[3][1,1]:=-0.15;
T[3][1,2]:=0.28;
T[3][2,1]:=0.26;
T[3][2,2]:=0.24;

T[4][1,1]:=0.85;
T[4][1,2]:=0.04;
T[4][2,1]:=-0.04;
T[4][2,2]:=0.85;



Tr[1][1]:=0; Tr[1][2]:=0;
Tr[2][1]:=0; Tr[2][2]:=1.6;
Tr[3][1]:=0; Tr[3][2]:=0.44;
Tr[4][1]:=0; Tr[4][2]:=1.6;

p[1]:=0.01;p[2]:=0.08;p[3]:=0.15;p[4]:=1;

end.
```

quel langage. Les codes des images que l'on veut reproduire ont été stockés dans une unité. Pour nos trois exemples, il s'agit de Unit Sierp pour le triangle de Sierpinski, Unit Foug pour l'image d'une fougère, Unit Arbre pour l'arbre fractal.

Explication de l'unité fougère

La partie Interface consiste en la définition des structures et des variables qui seront utilisées par le programme Fractal.

- **Ntrans :** le nombre de transformations de l'IFS.
- **Xechelle, Xoffset, Yechelle, Yoffset :** ce sont des constantes qui permettent d'ajuster le dessin à l'écran. Elles sont redéfinies en fonction de chaque écran et chaque image.
- **Mat** : c'est une matrice de dimension 2x2 Vec est un vecteur de dimension 2, Proba, un vecteur de dimension Ntrans qui contiendra les probabilités associées à chaque transformation.
- **Les variables :** T est un ensemble de Ntrans matrices, Tr (pour les translations) un ensemble de Ntrans vecteurs, et p une probabilité.
  La partie Implémentation consiste en l'initialisation des structures, transformations T, translations Tr et probabilités p.
  On remarquera que l'on donne des probabilités cumulatives pour simplifier le calcul du choix aléatoire de la transformation.

Explications du programme

- **Ligne 2** : le programme utilise les unités Graph pour le graphisme, Crt pour les fonctions mathématiques (ici random, mais également cosinus et sinus, si jamais les matrices sont proposées sous cette forme), et l'unité de l'image.
- **Ligne 3 :** les variables principales utilisées sont x,y nouvelX, nouvelY pour le calcul des coordonnées du point à tracer, et nbiter, le nombre d'itérations (c'est-à-dire le nombre de points qui seront tracés).
- **Ligne 4** : la procédure InitGraphisme pour initialiser l'écran graphique. Elle est classique et, après les vérifications d'usage, encadre l'écran d'un rectangle.
- **Ligne 5 :** la fonction Min permet de choisir une des transformations. Après le choix d'une valeur aléatoire x comprise entre 0 et 1, elle rend l'indice de la transformation dont la probabilité est la plus proche de x. Le fait que le vecteur p soit formé d'une suite croissante de réels raccourcit le calcul de cet indice.
- **Ligne 6 :** initialisation de l'écran graphique.
- **Ligne 7** : lancement du générateur de nombres aléatoires.
- **Ligne 8** : choix du nombre de points à tracer.
- **Ligne 9** : initialisation du premier point, ici à (0,0).
- **Ligne 10** : début de la boucle.
- **Ligne 11** : appel à la fonction Min: choix d'une transformation.
- **Ligne 12** : calcul des coordonnées du point transformé.
- **Ligne 15** : tracé du point; utilisation des échelles et d'une couleur (ici 14).
- **Ligne 16** : fin de la boucle.
- **Ligne 17** : attendre (pour admirer le dessin!)
- **Ligne 18** : retour à l'écran texte.
- **Ligne 19 :** fin du programme.

Référence de l'ouvrage : Michael Fielding Barnsley, Fractals everywhere, Academic Press.

LANGAGE

PC-LINT,

Vérification du style de programmation en C

FRANÇOIS BERGEON

L'utilisation de PC-Lint est similaire à celle du préprocesseur présent dans un compilateur C, à ceci près que la détection de certaines erreurs courantes de programmation s'ajoute à la simple vérification de syntaxe.

L'intérêt que représente un tel outil peut être mis en évidence en s'aidant de l'exemple classique suivant : en langage C, le test d'égalité est noté sous la forme de deux symboles "égal", ce caractère isolé est également utilisé dans le cas d'une assignation. Si, par malheur, un seul signe "égal" est présent dans ce qui devrait être une comparaison entre deux variables, le programme se comporte de manière incompréhensible, et la source du problème peut être très difficile à cerner. En effet la syntaxe "if (a = b)" n'est pas erronée en langage C, elle représente une recopie de la valeur de b dans la variable a, couplée à un test sur la valeur en question (vrai si différent de zéro).

Eviter les pièges classiques du langage C

De nombreuses autres constructions similaires déclenchent une erreur lorsque le programme est passé au tamis de PC-Lint. Au total, cet utilitaire dénombre plus de 270 messages d'erreur distincts, chacun se référant à un contexte bien précis, par exemple un problème de portabilité. Cinq types de messages sont traités par PC-Lint : les erreurs de syntaxe et les erreurs fatales (généralement détectées par les compilateurs également), les "warnings", les messages d'information, et les notes facultatives. Ces dernières ne sont pas affichées automatiquement, et requièrent le positionnement d'options afin d'obtenir leur apparition.

Le produit proposé par Gimpel Software offre une grande souplesse d'utilisation. En effet, l'affichage de certains messages d'erreur peut être purement et simplement supprimé par le programmeur, par exemple lorsqu'une partie de code utilise une syntaxe répertoriée comme dangereuse mais qui est, dans ce cas précis, néanmoins valide. De même, la tolérance générale de PC-Lint peut être augmentée ponctuellement lorsque le "header" d'une bibliothèque génère des erreurs de ce type.

Parmi les options offertes à l'utilisateur, citons un système permettant de définir l'aspect du message d'erreur généré, ainsi que l'indication de l'emplacement du "mot" fautif. Dans une utilisation courante, PC-Lint affiche par exemple la ligne de programme concernée par une erreur, une flèche pointant sur le caractère de cette ligne où l'erreur a été détectée, puis le message d'erreur lui-même.

Un autre paramétrage peut provoquer un affichage des messages d'erreur dans un format compatible avec le système de traitement des erreurs d'un éditeur de texte évolué (Brief, Epsilon, Sage, etc). PC-Lint est, de plus, paramétrable en fonction du type de compilateur utilisé afin de respecter ses spécificités ainsi que celles de ses bibliothèques.

Souplesse d'utilisation

Les différentes options de fonctionnement de PC-Lint se présentent sous la forme de "switches", soit directement insérés sur la ligne de commande, soit présents dans un fichier d'options, soit encore, à l'intérieur du programme source, encapsulés dans des commentaires. Cette dernière méthode permet de modifier dynamiquement les caractéristiques et la sensibilité de PC-Lint en fonction de la partie du programme considérée.

D'autres commentaires prédéfinis permettent de préciser que telle ou telle construction syntaxique est voulue par le programmeur, ceci afin d'éviter la génération d'un message d'erreur : "fall trough" dans le cas d'une continuité volontaire d'un bloc "case" vers un autre (absence de "break"); "notreached" pour confirmer que la suite du code ne sera pas exécutée; etc. Ces mots-clés sont en outre compatibles avec ceux supportés par le Lint d'Unix.

PC-Lint supporte des modules précompilés afin d'accélérer son exécution. Ainsi, un lancement préalable de PC-Lint sur les différents modules sources d'une bibliothèque permet de créer un fichier binaire qui pourra être utilisé lorsque PC-Lint vérifie le code d'un programme faisant appel à cette bibliothèque. Des modules préétablis pour les bibliothèques standard des compilateurs C les plus courants sont livrés avec le produit.

Par ailleurs, une option de PC-Lint permet de créer automatiquement un

La mesure de la qualité des sources d'un programme est une donnée hautement subjective. Il est, en revanche, possible de déceler de manière automatique la présence d'instruction inhabituelles, de constructions douteuses ou d'irrégularités d'écriture. Créé initialement sous Unix, l'utilitaire Lint permet de mettre en évidence les lignes de code qui contiennent un bug potentiel dans un programme en C.

fichier de prototypage des fonctions à partir d'un fichier source. Ce "header", ainsi créé, pourra être utilisé lors de la vérification d'autres modules au sein d'une même application.

Vivre avec Lint

Les premières utilisations de PC-Lint sont généralement assez décourageantes pour le développeur. Des fichiers sources, jusque là considérés comme "propres" par leur auteur, peuvent générer un nombre impressionnant d'erreurs Lint.

Un chapitre complet de la documentation de PC-Lint traite de cet état de fait en indiquant un exemple de paramétrage standard permettant rapidement à l'utilisateur de s'habituer au produit et à ses particularités. Ce guide d'utilisation contient un grand nombre de conseils pratiques qu'il sera sage de suivre. La suppression de certains messages d'erreurs trop contraignants sera par exemple préférable à l'abandon de la procédure de vérification...

En outre, l'utilisation de PC-Lint sur un projet important comprenant un grand nombre de fichiers sources se fait généralement étape par étape, en fait fichier par fichier. Ce n'est que lorsque les différents modules traités isolément ne génèrent plus d'erreur Lint qu'il est conseillé de vérifier l'application dans son ensemble. Cette dernière opération ne porte plus que sur les problèmes potentiels d'interfaçage entre les modules dans ce cas. A l'usage, PC-Lint s'avère très utile en décelant des problèmes avant qu'ils ne se déclarent en réalité.

La meilleure solution consiste en fait à avoir recours à PC-Lint dès les premières séances de codage, sans attendre d'avoir une version fonctionnelle de l'application. Dans cette optique PC-Lint peut être appelé directement à partir d'un fichier Make afin d'automatiser son exécution.

PC-Lint est proposé par Gimpel Software au prix de 139 dollars; la version spécifique 386 est, quant à elle, commercialisée au prix de 239 dollars pour une utilisation sur des projets nécessitant une grande quantité de mémoire. PC-Lint existe également sous le nom FlexeLint pour les systèmes Unix, OS-9, VAX/VMS, etc. □

Gimpel Software 3207 Hogarth Lane Collegeville, PA 19426 U.S.A tel: (19-1) (215)-584-4261 fax: (19-1) (215)-584-4266 .

EXEMPLE D'UTILISATION

Le programme ci-dessous est donné comme exemple dans la documentation de PC-Lint. Bien que sa compilation ne génère pas de message d'erreur de la part de la majorité des compilateurs, sa syntaxe est incorrecte, et l'affichage du résultat ne correspond pas à la valeur escomptée.

```
1       half(x)
2       double x;
3           {
4           return(x/2);
5           }
6
7       void main()
8           {
9           double n; 10 11     n =
half(10000); 12  printf("%d\n", n);
13          }
```

Lorsque ce programme est exécuté, le résultat affiché est 0 au lieu de 5 000. La cause de ce problème est triple : d'abord, la fonction half() n'est pas typée, et le compilateur la considère comme retournant un "int" par défaut et non un "double"; il faudrait écrire "double half(x)". Ensuite le paramètre passé à la fonction half() dans main() est un "int" et non un "double"; un double serait indiqué par "10000.00" par exemple. Enfin, le format utilisé par le printf() correspond à un "int" et non à un "double"; il aurait fallu utiliser "%f\n".

Compilé à l'aide de Borland C++ (warning maximum), seule l'absence de prototype pour half() est détectée à la ligne 11. PC-Lint, quant à lui détecte les anomalies suivantes :

- Ligne 2: pas de type explicite pour la fonction half(), "int"utilisé par défaut (message d'information).
- Ligne 4: perte de précision : "double" vers "int" (warning)
- Ligne 11: conflit de type d'argument, "int" au lieu de "double" (warning)
- Ligne 12: taille du second argument en désaccord avec le format du printf() (warning)

INSTALL,

Pour faire une bonne première impression

FRANÇOIS BERGEON

L'utilitaire Install de Knowledge Dynamics a pour but de simplifier la mise au point des procédures d'installation en mettant à la disposition du développeur un véritable langage de programmation dédié à cette tâche.

Bien au-delà des limites des fichiers "batch" classiques, une procédure Install est capable de gérer différentes options d'installation, de multiples disquettes de distribution, des fichiers compressés, des fichiers découpés et répartis sur plusieurs disquettes, etc. Par ailleurs, l'interface utilisateur particulièrement soignée est totalement transparente pour le développeur qui peut ainsi se concentrer sur les particularités d'installation de son logiciel.

Langage script spécifique

Afin de faciliter l'apprentissage du langage Install, cet utilitaire est accompagné de nombreux exemples de procédures qui couvrent la grande majorité des besoins usuels. Ce langage est d'un type très particulier puisque les sous-programmes qui composent un fichier script sont exécutés dans un ordre totalement prédéterminé et non modifiable. Par ailleurs, il propose des types de variables pour le moins inhabituels par exemple @Dir (nom de répertoire) et @Drive (unité de disque).

Un script Install est composé de blocs correspondant chacun à une phase prédéfinie de l'installation. Une procédure minimale peut être représentée, par exemple, à l'aide d'un bloc "produit" (@DefineProject) et d'un bloc "disque de distribution" (@DefineDisk), celui-ci contenant la définition des tâches à accomplir pour installer les fichiers présents sur une disquette (copie, décompactage, etc).

Ce script minimal peut être complété par des blocs d'affichage (@Display), de saisie du disque et du directory de destination (@GetOutDrive, @GetSubDir), de sélection des options d'installation (@GetGroups, @GetOptions), de modification des fichiers AUTOEXEC.BAT et CONFIG.SYS (@SetAutoexec, @SetConfig), ou encore d'actions à entreprendre en guise d'épilogue (@Finish).

Des variables prédéfinies permettent en outre de déclarer les paramètres généraux de l'installation : @Subdir et @OutDrive pour le répertoire et le disque de destination, @InDrive pour l'unité de disque source, @Name et @Version pour définir le logiciel à installer. Les constantes @OSMajor et @OSMinor représentent quant à elles la version du système MS-DOS utilisé. D'autres permettent d'identifier le type de processeur et de coprocesseur (@CPU, @NDP), la nature de la machine hôte (@MachineID), le type de carte vidéo et de moniteur utilisé (@VideoCard, @VideoMonitor), la quantité de mémoire conventionnelle ou EMS installée (@RAMTotal, @EMMTotal), la présence d'un clavier étendu (@Keyboard), ou encore celle d'une interface NetBios (@NetBios). Toutes ces constantes seront bien sûr exploitables à l'aide des blocs @If/@Elsif/@Else/@Endif disponibles dans le langage script. De plus, la présence d'une fonction de vérification d'existence d'un fichier (@Exist) permet de concevoir des scripts qui tiennent compte d'une éventuelle ancienne version du logiciel déjà présente sur le système.

Une vérification exhaustive

Install met en outre à la disposition du développeur une vérification de l'intégrité des fichiers installés grâce à un système de calcul et de comparaison de CRC. Il est également possible de regrouper plusieurs fichiers dans un seul fichier compressé sur les disquettes de distribution. Dans ce cas, le désarchivage de celui-ci est assuré de façon automatique par Install. Les fichiers de grande taille, quant à eux, peuvent être découpés et répartis sur plusieurs disquettes afin d'être reconstruits de manière totalement transparente lors de leur installation. Les utilitaires ARSENAL (archivage et compression) et SPLIT (découpage) permettent de préparer les fichiers en vue de leur distribution sous l'une de ces formes.

Install apporte une grande souplesse lors de la création des scripts d'installation en permettant au développeur de gérer indifféremment des disquettes de distribution de formats différents. Une seule et même procédure peut ainsi s'exécuter à partir d'un jeu de disquettes 360Ko, 720Ko ou 1,2Mo. Le bloc d'instructions correspondant à chaque disquette comporte en effet le nom de

L'installation est bien souvent le premier contact que l'on a avec un logiciel. De sa facilité de mise en place et de paramétrage dépend généralement l'attitude avec laquelle l'utilisateur aborde la phase de familiarisation avec le produit. Mais l'écriture d'une procédure d'installation simple, agréable, et efficace, demande un effort presque aussi important que l'écriture du logiciel lui-même...

tous les fichiers à installer présents sur celle-ci. Grâce à un système de "bibliothèque", il est aisé de faire correspondre le contenu d'une disquette 720 Ko à celui de deux disquettes 360 Ko par exemple.

Construction automatique

La génération des disquettes de distribution passe d'abord par la création d'un fichier DISK.ID présent sur chacune de celles-ci. Ce dernier contient, sous forme ASCII, le nom de l'application, son numéro de version, ainsi que le "label" disque tel que spécifié dans le fichier script (généralement identique à l'inscription portée sur l'étiquette de la disquette). La première disquette d'installation contiendra en outre une copie du programme INSTALL.EXE (distribué sous licence sans versement de royalties), ainsi que le script sous le nom immuable INSTALL.DAT (fichier texte).

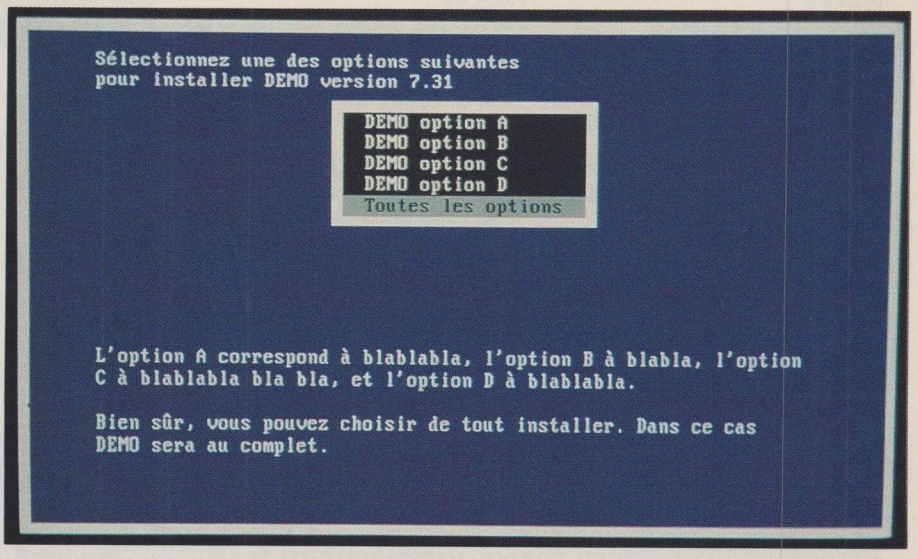
Install, dans sa version professionnelle, est avantageusement complété par l'utilitaire DDB. Le "Distribution Disk Builder" se présente comme une interface interactive à la création des disquettes de distribution en générant automatiquement le script Install correspondant. Le formatage des disquettes et la création des différents fichiers DISK.ID rentre également dans ses attributions. Le programme DDB peut être de plus utilisé pour créer une nouvelle version d'un fichier script et d'un jeu de disquettes de distribution.

L'un des reproches que l'on pourrait faire à Install est que le programme démarre par l'affichage d'un Copyright mentionnant Knowledge Dynamics, ce qui ne sera pas forcément du goût de tous les développeurs. Par ailleurs, le fichier d'installation n'est nullement pré-compilé, ni même codé, et l'utilisateur peut donc avoir connaissance de son contenu (le fichier INSTALL.DAT présent sur les disquettes du produit Install est d'ailleurs un exemple supplémentaire à mettre à son actif). En outre, aucune option n'a été prévue pour exécuter un fichier script qui ne s'appellerait pas INSTALL.DAT.

Notons enfin que Install et DDB sont fournis avec leurs sources complets en C et les fichiers Make correspondants. Cette particularité permet par exemple d'adjoindre une fonctionnalité supplémentaire faisant défaut, ou encore de traduire les messages affichés par le programme. Une option internationale d'Install est cependant disponible afin d'éviter ce travail au développeur. Ajoutons qu'une option OS/2 est également disponible.

En permettant la mise au point rapide de procédures d'installation agréables et efficaces, Install s'avère être un outil de choix. Il sera utile, non seulement au développeur visant à créer un logiciel commercial, mais également au programmeur qui doit distribuer de nombreuses copies de son logiciel (ou d'une nouvelle version de celui-ci) au sein de l'entreprise. L'investissement sera généralement largement amorti par le gain de temps réalisé à la fois par l'utilisateur et l'installateur.

Knowledge Dynamics Highway Contract 4 Box 185-H Canyon Lake, TX 78133-3508 U.S.A tel: (19-1) (512)-964-3994 fax: (19-1) (512)-964-3958



INFO PC

LE MAGAZINE DE LA MICRO-INFORMATIQUE PROFESSIONNELLE

LANGAGES ET SYSTÈMES

Immeuble La Fayette, 2, place des Vosges, Cedex 65, 92051 Paris La Défense 5, Tél. : 49.04.79.00 Télex : 613 234 F MONDINF

DIRECTEUR : JACQUES TILLIER assisté de CATHERINE BOISADAM,

RÉDACTEUR EN CHEF : PHILIPPE MONNIN, **ASSISTANT DE LA RÉDACTION :** STÉPHANE GAUTIER,

1ère SECRÉTAIRE DE RÉDACTION : CATHERINE OLZCHANECH,

SECRÉTAIRES DE RÉDACTION : ELISABETH DÉROFF, MICHÈLE DOUAY, ISABELLE DUMONTEIL, STÉPHANIE GRIN

CE NUMÉRO A ÉTÉ RÉALISÉ SOUS LA RESPONSABILITÉ DE YVES GUITTARD

Ont également collaboré à ce numéro : Marc Adler, François Bergeon, Jean-François Duvivier, Luc Lemaire, Henry Lemberg, Atta Oloumi, Henry-Luc Planchat, Matthieu Thfoin, Christophe Souchard.

Les articles et illustrations publiés en page 10 sont repris après l'accord de Microsoft Corporation, Redmond, Washington, USA.

Copyright@ 1990 par Microsoft Publications. Tous droits réservés.

La rédaction d'INFOPC dégage sa responsabilité en ce qui concerne les documents écrits, les matériels et les logiciels qui lui seraient envoyés sans demande préalable de sa part.

PUBLICITÉ : DIRECTEUR DE LA PUBLICITÉ : SOPHIE TALAYRACH, **DIRECTEUR ADJOINT DE LA PUBLICITÉ** : NATHALIE TESSLER,

CHEFS DE PUBLICITÉ : VIRGINIE BRIL, THIEN GILBERT, HERVÉ LABATTUT, **ASSISTANTE DE PUBLICITÉ :** AROMI RAJON

DIFFUSION : DIRECTEUR : JEAN PAITEL, **RESPONSABLE SERVICE ABONNEMENTS** : LUCIENNE BOSSER assisté de JOSIANE LIBAN, ROSE MOLNAR, ALAIN PACCAGNELLA, BERNADETTE THUAL, **RESPONSABLE DES VENTES** : PIERRE-OLIVIER PROVENT

Tél. : 49.04.79.68, Abonnements : 49.04.79.67, 49.04.79.72, Suisse : MADITEC SA - Case postale 144 CH1020 RENENS 1 - Tél. : (021)35.08.65, Canada et Etats-Unis : Editions Gosselin Inc. 228, rue de la Colline, AYLMER PQ j9j 1t8, CANADA - Tél. : 819-684-7885

INFOPC est une publication éditée par Micropresse S.A. Capital de 250000 F, en location-gérance auprès de

IDG COMMUNICATIONS FRANCE

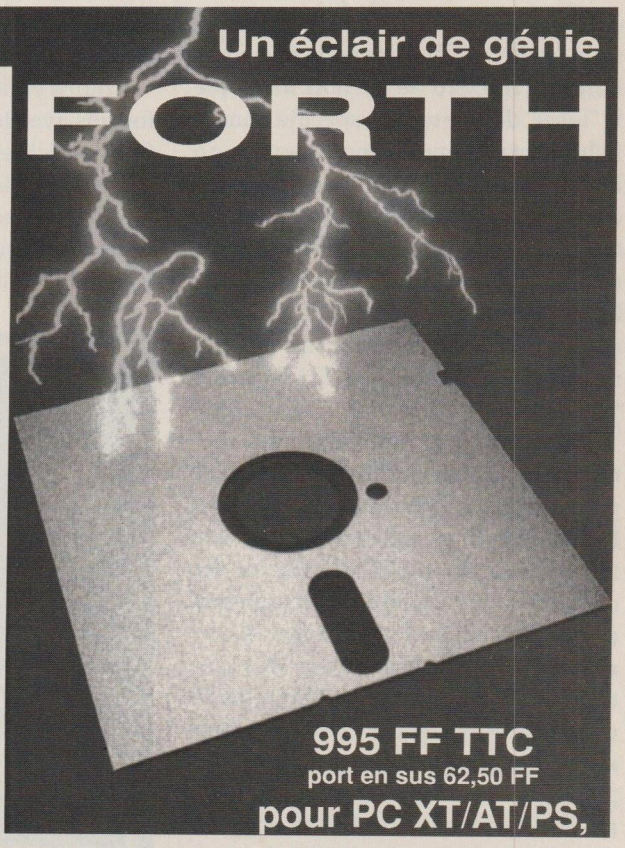
ISSN 0981 - 6402

Service lecteurs n° 8 (dernière page)

DANS LA JUNGLE DES LANGAGES DE DEVELOPPEMENT, VOICI DES RAISONS DE FAIRE LE BON CHOIX

TURBO-FORTH

Un éclair de génie



VOUS OFFRE:

- à la fois un interpréteur et un compilateur, fonctionnant EN UNE SEULE PASSE pour IBM PC/XT/AT/PS2, et Atari Portfolio, ce sans nécessiter l'utilisation de code objet et de linker!!!
- un vrai multi-tâche sous DOS,
- un métacompilateur permettant de créer des exécutables hyper-compacts,
- un débogueur-traceur permettant de suivre pas à pas les procédures compilées et le contenu de la pile de données,
- un décompilateur capable également de désassembler les procédures rédigées directement en code machine.
- un assembleur structuré utilisant de véritables mnémoniques, permettant la définition de nouvelles macro-instructions par l'utilisateur,
- trois modules de calcul virgule flottante (y compris un modèle CORDIC)
- une auto-documentation intégrée et contextuelle,
- un éditeur offrant des fonctions traitement de texte,
- le traitement hiérarchisé des fichiers sources ASCII permettant huit niveaux d'imbrication, ainsi que l'interprétation ré-entrante des chaînes alphanumériques (e.g. exécution du contenu de cette chaîne)
- l'appel aux programmes externes: C, Pascal, etc...
- de nombreux utilitaires: accès aux fichiers dBase, pilotage des ports d'entrée/sortie, gestion des modes graphiques, accès aux variables d'environnement DOS, création de programmes résidents.

Produit français, TURBO-FORTH vous offre près de 1 Mo de fichiers sources et exécutables. Il traite directement les fichiers ASCII, que ce soit en édition, en interprétation, et en compilation. En phase de test, chaque procédure peut être exécutée indépendamment du programme général. Ce langage de quatrième génération fonctionne sous DOS (2.0 et suivants) et sous CONCURRENT DOS 386. Il est accompagné de deux manuels (manuel d'apprentissage et guide de référence).

Configuration minimale requise : 128 Ko RAM (64 Ko pour la version ATARI Portfolio)!!!!

995 FF TTC

port en sus 62,50 FF

pour PC XT/AT/PS,

REM CORP - 17, rue de la Lancette - 75012 PARIS (France)

tel: (33) 1- 43.42.32.15 fax: (33) 1- 43.40.96.53 Minicom: 3612 REM CORP 143409653 Calvacom: cité FORTH

C-scape : un véritable plaisir.

Gestionnaire d'interfaces écran graphique et texte

Le système de gestion d'écrans C-scape® est une bibliothèque de fonctions en langage C permettant la saisie et la validation de données à l'écran, la gestion de menus et fenêtres, l'édition de texte, et les systèmes d'aide en ligne. L'outil de conception d'écrans de C-Scape, Look & Feel™, vous permet de créer des écrans de façon interactive et de générer le code source correspondant.

C-Scape comprend des fonctions de haut niveau facilement modifiables, aussi bien que des primitives permettant de construire de nouvelles fonctions. Grâce à l'architecture orientée objet de C-Scape, vous développerez des applications plus fonctionnelles, plus flexibles, plus portables, et vous aurez plus de plaisir à le faire.

Un produit phare dans l'industrie du logiciel. Depuis 5 ans, plusieurs milliers de développeurs dans le monde entier se sont convertis à C-Scape, abandonnant leur bibliothèque "maison", ou des produits peu flexibles. La presse approuve: "C-Scape est de loin le meilleur. ...un véritable plaisir", ecrit IEEE Computer. Les plus grandes sociétés ont selectionné C-Scape comme standard de développement.

C-Scape peut être utilisé avec des SGBD, des bibliothèques graphiques, ou toute bibliothèque C ou C++. De plus, C-Scape peut combiner texte et graphique. Pour porter votre application de DOS ou OS/2 vers UNIX, VMS, AIX, recompilez simplement. C-Scape supporte également les "DOS extenders" Phar Lap, Rational et Ergo, ainsi que MetaWINDOW.

Graphique. Fonctionne en couleur, en mode texte ou graphique. Peut lire des images graphiques depuis des fichiers PCX.

Architecture orientée objet. Ajoutez vos propres fonctions et créez des modules de code réutilisables.

Support de la souris complètement intégré pour la selection des options de menu, les champs de saisie, et pour déplacer et redimensionner une fenêtre.

Portabilité. Le code de C-Scape est indépendant du hardware. Supporte DOS, OS/2, UNIX, AIX, VMS, QNX, XENIX... Détecte automatiquement Hercules, CGA, EGA, VGA, et les modes 43 et 50 lignes.

Edition de texte. Comprend un éditeur de texte avec contrôle de la césure, fonctions rechercher/remplacer, et manipulation des blocs de texte.

Flexibilité des champs. Champs masqués, protegés, marqués, avec label, obligatoires, avec ou sans echo. Validation complète des données. Champs de type heure, date, montant, liste déroulante, et autres fonctions de haut niveau. Créez votre type de champ.

Fenêtres. Encadrées, pop-up, superposées, eclatées, ascenceur, défilement horizontal; taille et nombre des fenêtres limités par la memoire RAM uniquement.

Menus. Déroulants, hiérarchiques, pop-up, style Lotus 1-2-3, ou créez le vôtre.

Système d'aide contextuel. Prompts et messages d'aide attachés à un écran ou à un champ. Messages à références croisées pour créer un système d'aide en ligne de style hypertexte.

Gestion des écrans. Ecrans stockés dans des fichiers ou linkés avec l'application.

Génération de code. Créez n'importe quel écran avec l'outil interactif Look & Feel, testez-le, puis convertissez-le automatiquement en code C.

Essayez C-Scape ! C-Scape est *puissant, flexible, portable*, d'apprentissage et d'utilisation facile. C-Scape est livré en standard avec la totalité du code source de la bibliothèque et l'outil de création d'écrans Look & Feel. La documentation est claire, extensive et detaillée. Avec C-scape, pas de module résident et pas de royalties sur les applications diffusées.

C-Scape est un produit d'**Oakland Group**. Smartline assure le support technique de C-scape par téléphone et par fax ainsi que des sessions de formation sur le produit.

Appelez-nous au (1) 46 34 64 19
ou envoyez le bon de commande ci-dessous par fax ou par courrier. Après le plaisir de C-Scape, le développement ne sera jamais plus ce qu'il était.

C-scape et Look & Feel **version 3.2** : DOS, OS/2 (Borland, Microsoft, Zortech): 4500FHT; UNIX: 13500FHT; Autres OS: nous consulter. Cartes de crédit acceptées.

smartline

30, rue Saint André des Arts
75006 Paris France
Fax: (1) 40 46 85 89 ***Tél: (1) 46 34 64 19***

Oakland Group, Inc 675 Massachusetts Ave, Cambridge, MA 02139 USA.Tél: 617-491-7311. Les produits d'Oakland sont distribués en Australie, Belgique, Danemark, Autriche, Allemagne, Suisse, Norvège, Suède, Angleterre, France. La photographie représente un programme C-Scape combinant une saisie de données avec des images vidéo chargées depuis un fichier .PCX. C-Scape et Look & Feel sont des marques déposées par Oakland Group, Inc.; les autres marques citées appartiennent à leurs propriétaires respectifs. Les fonctionnalités, prix et conditions sont sujets à modification. Documentation en anglais. C-scape (seascape): paysage de mer.

Service lecteurs n° 9 (dernière page)

❑ Je désire recevoir une documentation sur C-scape

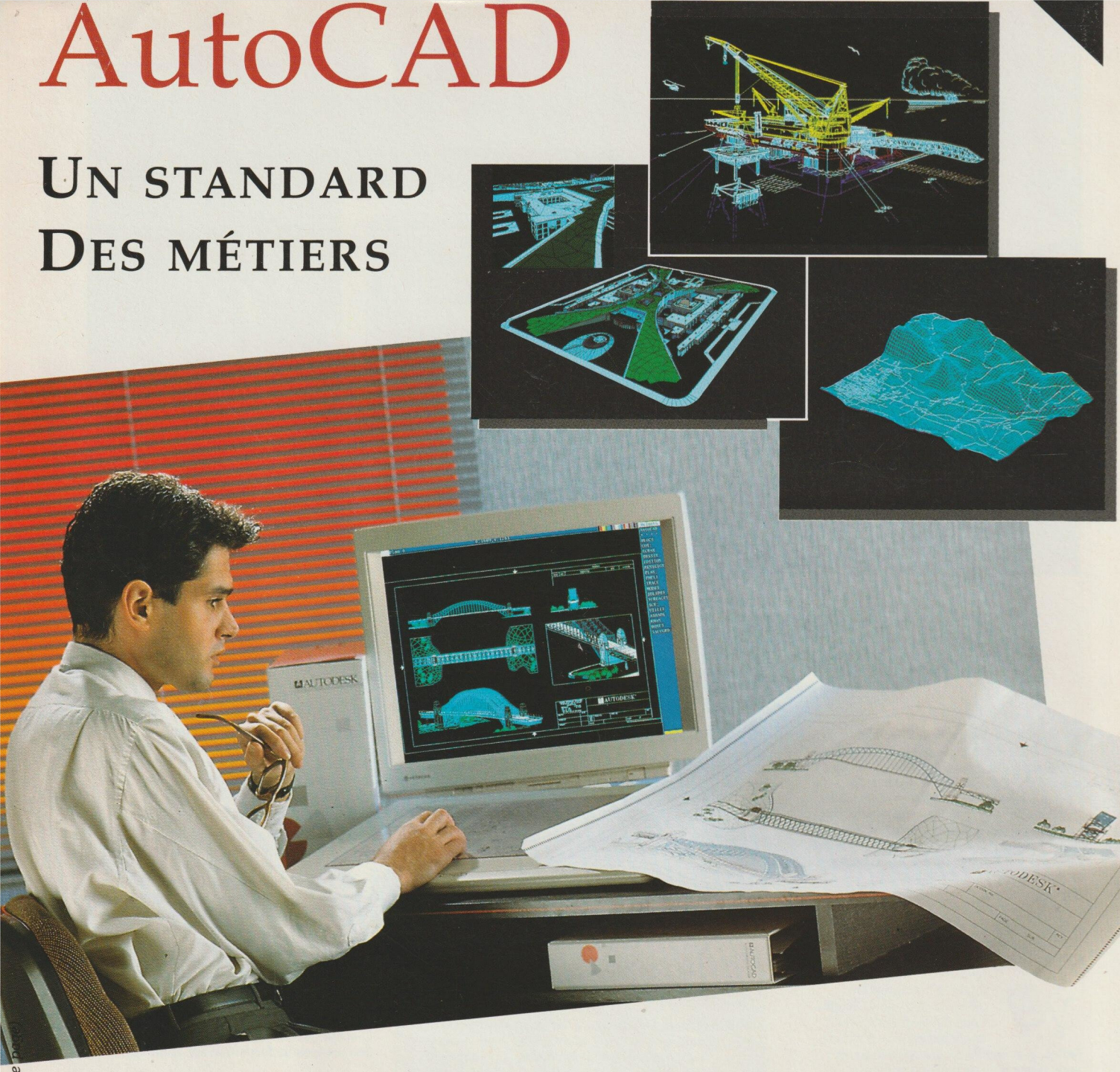
❑ Je désire recevoir le dossier d'évaluation comprenant les deux disquettes de démonstration C-scape et Look & Feel. Ci-joint un chèque de 120 F TTC. Une facture sera jointe à l'envoi.

nom prénom _____ société _____ tél _____

adresse _____

AutoCAD

UN STANDARD DES MÉTIERS



1991 **L'année de nos 500 000 utilisateurs !**
25 000 recencés en France dans tous les secteurs d'activité : cartographie, architecture-bâtiment, génie électrique, mécanique, ...
AutoCAD est le standard des logiciels de CAO.

La version 11 d'AutoCAD vous offre plus !
Plus de productivité : tracés simultanés de vues multiples, accès à la modélisation solide avec le module AME, ...
Plus de flexibilité et d'ouverture : nouvel environnement de développement ADS en langage C, compatibilité totale entre vos fichiers générés sous micro-ordinateurs et stations de travail.

La commercialisation par un réseau de revendeurs agréés !
Nos revendeurs AAD (AutoCAD Authorized Dealer) vous assureront le meilleur service.

AUTODESK, leader mondial des logiciels de CAO vous garantit la pérennité de votre investissement.

AUTODESK - 7, cours des Roches - BP 312 Cedex 2 Noisiel - 77443 Marne-la-Vallée
Tél. : (1) 60 17 80 55 - Fax : (1) 60 17 80 18

AUTODESK

Service lecteurs n° 10 (dernière page)